Defence Research and Development Canada | Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

CODEM User Manual

François Rioux, Jean-Philippe Gagnon

LTI
2700, De Carthagène,
Québec, Qc
Canada
G2B 5M4

Scientific Authority: M. B. DuCharme (418) 844-4000 Ext.: 4224

Defence R&D Canada – Valcartier
Contract Report
DRDC Valcartier CR 2010-518
September 2010

Canada

LTI Informatique & génie Software & engineering

CODEM User Manual

September 2010

Prepared By

Fran�ois Rioux � LTI

Jean-Philippe Gagnon � LTI

Prepared For

Michel B. DuCharme, Daniel Lafond and Michel Lizotte

DRDC - Valcartier

IMAGE-SCE

Authors

Fran�ois Rioux � LTI

Jean-Philippe Gagnon � LTI

Reviewed by

Daniel Lafond � DRDC - Valcartier

2700, De Carthag�ne,
Qu�bec, Qc
Canada
G2B 5M4

Website: www.ltinfo.ca
E-mail: contact@ltinfo.ca

Release date :	September 2010
Category :	Final Project Report
Version :	2.1
Comments :	

Abstract

This report details the implementation of an experimental platform that will be used in order to study how humans understand complex systems / situations. It implements a microworld as well as a scenario building tool that facilitates the development of a range of scenarios using a common platform. A user guide of the platform is first provided in this report. It describes how to use the user interface that enables interaction with the situation and how to configure its various parameters. The scenario building tool is also presented. It allows adding data structures to a scenario that will enable/disable associated functionalities. It exploits a simulator that was built as part of the IMAGE project in order to simulate system dynamics, the formalism that was chosen to represent a complex situation. Finally, additional technical details are provided in the appendices.

R�sum�

Ce rapport d�crit l�implantation d�une plateforme exp�rimentale visant � �tudier la mani�re dont les humains arrivent � comprendre des syst�mes / situations complexes. Le logiciel d�velopp� implante un micromonde ainsi qu�un outil qui facilite le d�veloppement de sc�narios divers � l�aide d�une plateforme unique. Ce rapport fournit tout d�abord un guide de l�utilisateur de la plateforme exp�rimentale. Il d�crit comment l�interface utilisateur devrait �tre utilis�e ainsi que la mani�re de configurer les diff�rents param�tres qui rendent cette plateforme si flexible. L�outil de g�n�ration de sc�narios est par la suite d�taill�. Il permet l�ajout de structures de donn�es dans un sc�nario qui viendront activer/d�sactiver les fonctionnalit�s associ�es. Cet outil exploite un simulateur d�velopp� dans le cadre du projet IMAGE, afin de simuler la dynamique des syst�mes, le formalisme choisi afin de simuler des situations complexes. Finalement, plusieurs d�tails techniques sont fournis en annexe.

This page is intentionally left blank.

Executive Summary

The current experimental platform will allow experimenters to study how humans come to understand complex systems / situations. Participants need to complete mandates that involve one or more human participants who have to make decisions on possible interventions in order to steer the system (or situation) to a desired state. For that purpose, a highly configurable and generic user interface was built. It allows for a participant to perceive the current situation, inspect influence diagrams describing the known cause-effect relationships in the system, make predictions about what the situation will look like in the next turn, make decisions by distributing action points on interventions, reading information snippets and taking notes on what the current strategy is. In addition to the latter content, the user interface provides useful feedback at the end of a turn and at the end of a mandate and displaying information about the current state on each turn. Several participants at a time can contribute to the evolution of a simulated situation, either in collaboration or in competition. The present document constitutes a complete user guide on how to use the platform effectively.

This report first introduces how to use the user interface associated to the experimental platform. It was built in such a way that it supports the arrangement of visual components associated to an arbitrary number of variables and interventions; the layout of the components will be automatically updated according to the elements specified in the scenario editor. It is thus completely generic, not tied to a given scenario. The platform also provides software hooks that allow for additional user-defined features, two of which are provided as examples. The appendices of this report contain technical details regarding the content of user logs and methods that should be overridden in order to implement a software hook.

The last sections of this report detail how to develop a scenario from scratch. For the purpose of facilitating scenario building and avoiding source code development by users who are not proficient at it, a scenario editor was developed. This editor reads a configuration file that defines every single possible parameter (i.e. name, type, description) in different parts of simulation entities, and then fills a user interface with the parameter values found in a scenario file. Parameters can then be modified, variables and action points created / removed, making it possible for users to build a scenario that will be written directly to a CODEM scenario file (□cdm□ extension), and loaded by a simulator thereafter.

Sommaire

La plateforme expérimentale développée dans le cadre de ce projet permettra à un expérimentateur d'étudier comment les humains en arrivent à comprendre des systèmes / situations complexes. Les participants doivent compléter des mandats impliquant un ou plusieurs participants qui ont à prendre des décisions sur des interventions possibles afin d'amener la situation complexe vers un état désiré. Afin de supporter les expérimentateurs dans leurs recherches sur la prise de décision en situation complexe, une interface utilisateur des plus flexibles a été développée. Elle permet aux participants de percevoir l'état courant de la situation, d'inspecter le diagramme d'influence décrivant les relations de cause à effet connues dans un système, de prédire l'état de la situation au tour suivant, de prendre des décisions en allouant des points d'actions sur différentes interventions, de lire des informations pertinentes sur l'état de la situation, ou de prendre des notes sur la stratégie adoptée. En plus des fonctionnalités précédentes, l'interface fournit du feedback à la fin de chaque tour complété tout en gérant adéquatement la collaboration / compétition entre plusieurs participants. Le présent document est un guide de l'utilisateur complet sur comment utiliser la plateforme de manière efficace.

Ce rapport introduit tout d'abord l'interface utilisateur associée à la plateforme expérimentale. Cette interface a été conçue d'une manière telle qu'elle supporte le réarrangement automatique de ses composantes graphiques en fonction du nombre de variables ou interventions présentes dans le système. La plateforme définit également des interfaces logicielles qu'il est possible d'implanter afin d'ajouter des fonctionnalités pour un scénario donné. Deux exemples de telles fonctionnalités sont fournis dans ce rapport. Finalement, les annexes fournissent des détails supplémentaires sur le contenu des fichiers de log ainsi que sur les méthodes qui doivent être implantées afin de bien utiliser les extensions logicielles.

Les dernières sections de ce rapport détaillent un éditeur graphique qui permet la création de scénarios de manière générique tout en assurant la validité du contenu ajouté par un utilisateur. En lisant un fichier de configuration, cet éditeur conna’t tous les paramètres que peut contenir un modèle, et les rend disponibles à l'utilisateur. Ce dernier peut donc modifier les paramètres à sa guise et sauvegarder le résultat de son travail dans un fichier de scénario CODEM (extension « cdm ») qui peut être chargé directement dans un simulateur. Le développement d'un scénario complet en est ainsi facilité.

Table of Contents

Abstract i

Résumé i

Executive Summary ii

Sommaire iii

Table of Contents iv

List of Figures vi

List of Tables viii

1 Introduction 1

2 User Guide of the Experimental Platform 2

2.1 Situation Tab 3

2.2 Relations Tab 4

2.3 Prediction Tab 6

2.4 Decision Tab 7

2.5 Info Tab 8

2.6 Notes Tab 12

2.7 End of Turn Feedback Screen 13

2.8 Setting the Platform to Support Multiple Participants 15

2.9 Log Files 16

2.10 Language Configuration 16

3 Scenario Editor's User Guide 17

3.1 Generic Scenario Parameters 18

3.2 Description of Variables 23

3.2.1 Mutual Influences 26

3.2.2 Conditional Mutual Influences 28

3.3 Description of Action Points 30
3.3.1 Interventions 31
3.3.2 Variable Influence on Action Points 34
3.3.3 Player-Specific Representation of Variables 35
3.4 Description of Custom Events 36
3.5 Description of Information Text 37
3.6 Configuration of the Scenario Editor 39
3.7 Extensions 39
3.7.1 Insurgents Strategy Extension 41
3.7.2 Stable Prediction Error Extension 41
3.7.3 Genetic Algorithms for Finding Optimal Solutions 42
4 Conclusion 43
5 References 44
A. Log Files Content Description 45
A.1 Participant s Log Description 45
A.2 Game s Log Description 51
A.3 Summary Logs 53
B. Text Labels of the User Interface 54
C. Details of the UIDispatchAdapterI Interface 58
D. Quick Start Reference Guide 64
D.1 Walkthrough 64
D.1.1 Single User 64
D.1.2 Multiple Users (Cooperative or Competitive) 64
D.2 Important Files 65
D.3 Important Operations in the Experimental Platform 66
D.4 Troubleshooting of the Experimental Platform 66
E. From Conceptual Representations to System Dynamics Simulation 68
E.1 Conceptual Representation with CoGUI 68
E.2 Agent-Based Simulation Using Conceptual Representations 69
E.3 Agent-Based System Dynamics Simulation 71

List of Figures

Figure 1 - Situation Tab....4
Figure 2 - Relation Tab....5
Figure 3 - Chart of Influence....6
Figure 4 - Prediction Tab....7
Figure 5 - Decision Tab....8
Figure 6 - Info Tab (Messages)....9
Figure 7 - Info Tab (Diagram)....10
Figure 8 - Info Tab (History)....11
Figure 9 - Info Tab (Feedbacks)....12
Figure 10 - Notes Tab....13
Figure 11 - Feedback screen (End of turn Summary)....14
Figure 12 - Feedback Screen (Prediction Error)....15
Figure 13 - Scenario parameters "Main Dialog Parameters" group....18
Figure 14 - Scenario parameters "End of Game Parameters" group....18
Figure 15 - Scenario parameters "Game Mode Parameters" group....19
Figure 16 - Scenario parameters "Transition and End of Mandate Parameters" group....19
Figure 17 - Scenario parameters "Instances" group....23
Figure 18 - Variable addition and removal controls....23
Figure 19 - Simple variables parameters....24
Figure 20 - Details of the color categories for variables values....24
Figure 21 - Sample chart for the influence editor....26
Figure 22 - Variables' mutual influences parameters....27
Figure 23 - Variables' conditional mutual influences parameters....29
Figure 24 � Action points addition and removal controls....30
Figure 25 - Action points base parameters....30
Figure 26 - Interventions parameters....32
Figure 27 � Variable influence on action points....34
Figure 28 � Player-specific representation of variables....35
Figure 29 - Custom events parameters....36
Figure 30 - Info text parameters....38
Figure 31 - Insurgents strategy parameters....39

Figure 32 - Mean prediction error as a function of turn number showing the prediction error's slope in a time window 42
Figure 33 - Example of a conceptual graph in CoGUI 69
Figure 34 - Scenario generation process in IMAGE 71
Figure 35 - Conceptual graph hierarchy from the experimental platform 73
Figure 36 - Sequence of behaviours executed in the custom agent-based simulator 74

List of Tables

Table 1 - Detailed parameters of a scenario 19
Table 2 - Details of simple variables parameters 24
Table 3 - Details of "Influences" parameters 27
Table 4 - Summary of Javascript operators 29
Table 5 - Details of parameters associated to Action Points 30
Table 6 - Details of basic Interventions parameters.......... 32
Table 7 - Details of parameters associated to Intervention Effects 33
Table 8 - Details of variable influence on action points parameters.......... 34
Table 9 - Details of the Custom Events parameters 37
Table 10 - Details of Info Text parameters 38

This page is intentionally left blank.

1 Introduction

The complex decision making experimental platform, CODEM, is a generic and flexible test bed that is intended to be a shareable research tool to stimulate coordinated and synergetic research on complex dynamic problem solving. One of the formal requirements about the current experimental platform is that a typical scenario should run without the need for the user (either an experimenter or a participant) to write a single line of source code. Due to this requirement, the experimental platform is constrained to allow the development of a specific class of models called system dynamics models (Sterman, 2000), which define a situation using numerical variables and where influences between variables are positive or negative numerical effects. Additional details will be provided in the following sections.

The current experimental platform involves one or more human participants that have to make decisions on possible interventions in order to steer the system (or situation) to a desired state. For that purpose, a highly configurable and generic user interface framework was built. It allows for a participant to perceive the current situation, inspect influence diagrams describing the known cause-effect relationships in the system, make predictions about what the situation will look like in the next turn, make decisions by distributing action points on interventions, reading information snippets and taking notes on what the current strategy is. In addition to the latter content, the user interface framework provides useful feedback at the end of a turn and at the end of a mandate while adequately displaying the current state of a turn. Several participants at a time can contribute to the evolution of a simulated situation, either working collaboratively or competitively. The present document constitutes a complete user guide on how to use the platform effectively. A quick start reference guide is provided in Appendix D.

A scenario editor was built in order to facilitate the development of a *specific* scenario, reading and writing directly CDM scenario files. In order to avoid the need of writing Java source code, a generic software architecture was put in place, which allows for arbitrary number of participants, variables and interventions to be defined.

2 User Guide of the Experimental Platform

The main goal of CODEM is to study how humans deal with complex situations. The simulated scenarios take the form of a turn-based strategy game, similar to previous work on complex problem solving (D�rner, 1996). Therefore, a user interface controlling the progress of a game was built and integrated to the platform. It has the following objectives:

- Provide the user with a view of the current situation;
- Show to the user the relations between the variables describing the situation;
- Allow the user to make predictions about the state of the situation in the next turn;
- Allow the user to influence the current situation by making decisions that consist of allocating action points on various interventions;
- Allow the user to retrieve information about the situation and the decisions history;
- Allow the user to take notes during his mandate;
- Provide the user with a feedback on variables changes that occurred during a turn and on the provenance of these changes, as well as his prediction error.

The experimental platform�s user interface was built in such a way that it supports the arrangement of visual components associated to an arbitrary number of variables and interventions; the layout of the components will be automatically updated according to the elements specified in the scenario editor. Most of the features can also be enabled and/or disabled via configuration parameters available in the scenario editor (see Section 3.1 for complete details).

Prior to starting CODEM, it is recommended to register file extensions �.cdm� to the CODEM executable, which will allow starting the platform via a double-click on a valid scenario file (�.cdm� extension). In order to do so, simply open the scenario editor via �CODEM_ScenarioEditor.bat� and choose in the menu bar �Config / Register CDM files�. If the registration succeeds, at computer restart, �.cdm� files will be associated to a new icon that is the CODEM logo. Double-clicking on such a file will launch CODEM. Note that scenario files are stored in the �data� directory. Just after CODEM is started and when the □Identify User□ parameter is set, the platform asks to identify the participant, session and group IDs. Those IDs are included in the name of the log files, located in the □log□ directory.

In the main user interface, five different views are available via a tabbed pane: situation, prediction, decision, information and notes. Features of the experimental platform have been separated in five views due to the need of recording in a log file how much time users spend doing specific subtasks (situation assessment, inspection of mutual influences, note taking, etc.) while completing their mandate.

A view that provides users with feedback at the end of a turn is also available. It will show to the user the evolution of the situation during the last turn and the prediction errors he made. Finally, a view that summarises the entire mandate history can be provided at the end of a mandate.

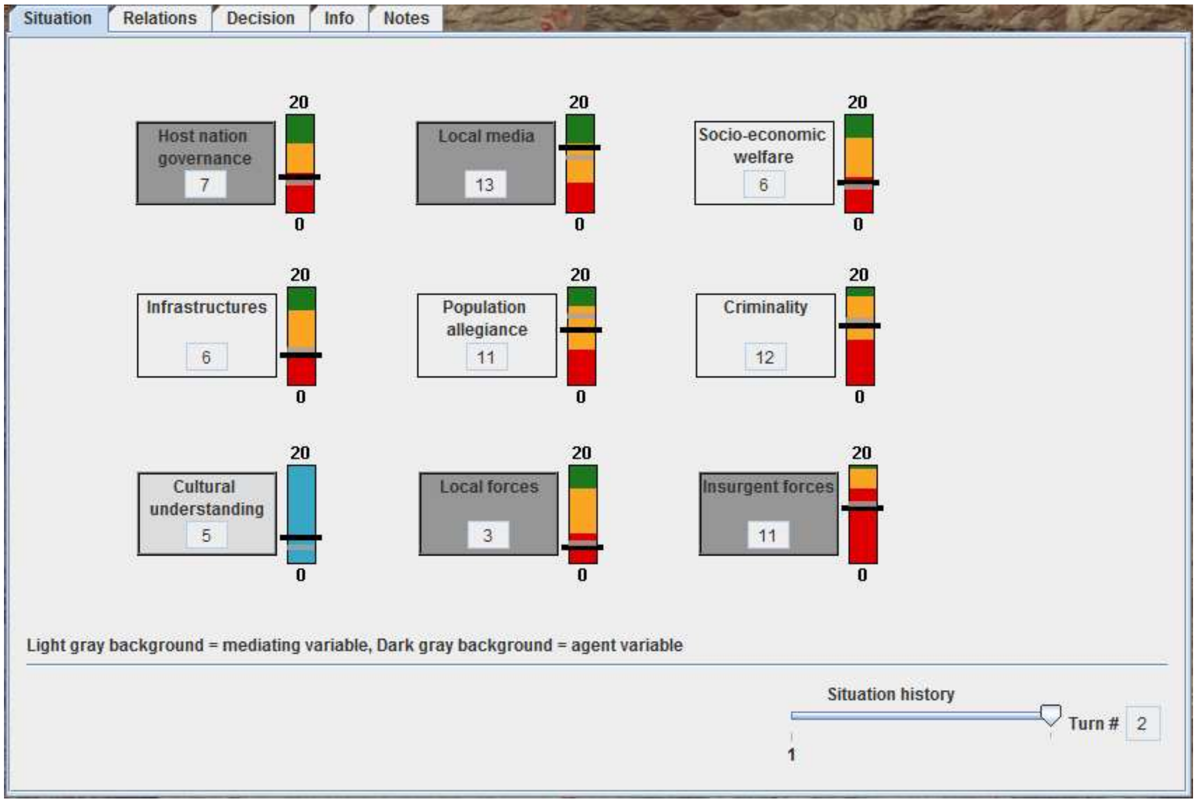
It should be noted that several indications can be shown to the user during the mandate regarding the status of the simulator. For example, when waiting for other participants, an image is displayed that informs the user about the current status of the system; at the end of a mandate, images can be displayed in order to inform the user as to whether he succeeded or failed in completing the mandate. A user will not be able to close any window that would terminate the simulation without reaching the end of a mandate. There is only one way to terminate a mandate prematurely: pressing the **CTRL+ALT+X** keys simultaneously. Doing so is equivalent to letting the situation evolve without allocating any action point to interventions.

In order to fully customize the user interface of the experimental platform, the placement of the variables□labels can be changed by the user. Hitting the □CTRL+E□ key switches from □normal□ mode to □editing□ mode. In editing mode, variables□labels can be moved around the user interface by dragging them with the mouse. The main dialog can also be resized; for that purpose, simply move the mouse cursor to the edge of the dialog. A □resize□ cursor will appear. Then press the left mouse button and drag the edge of the dialog to the desired size. Hitting □CTRL+E□ will save the new layout and size to a file (default is □config/layout.xml□, but it can also be configured in the editor), which will be reloaded next time the experimental platform is executed.

The following sections present in detail how the interface should be used. The reference scenario is the prototype of a complex counterinsurgency situation developed at DRDC Valcartier that includes nine variables and in which the user can affect the situation via seven interventions.

2.1 Situation Tab

The *Situation Tab* (see Figure 1) displays the status of all variables that are involved in a situation. In order to win the game in the context of the counterinsurgency scenario, the value of all variables must be outside of the red region displayed on coloured indicators. The current variable□s value is indicated graphically using a horizontal black line while the previous turn□s value is shown using a gray line. As turns pass, the user can go back in time in order to examine the values of previous turns using the slider positioned at the bottom of the tab. When the situation tab does not display the current turn, it is greyed out in order to indicate that it does not show values associated to the latest turn. Depending on the game configuration parameters, the user is asked to make his predictions on the next variables□ values, or to make decisions on interventions, or both, before he can continue to the next turn by pressing the button located in the tab corresponding to the last operation (decision or prediction). This latter button is only enabled when mandatory predictions and/or decisions have been made. It should be noted that when the platform is configured with a time-limit for each turn, the turn advances automatically when the time limit is reached, even though predictions and decisions have not entirely been completed.



Host nation governance 7
Local media 13
Socio-economic welfare 6
Infrastructures 6
Population allegiance 11
Criminality 12
Cultural understanding 5
Local forces 3
Insurgent forces 11

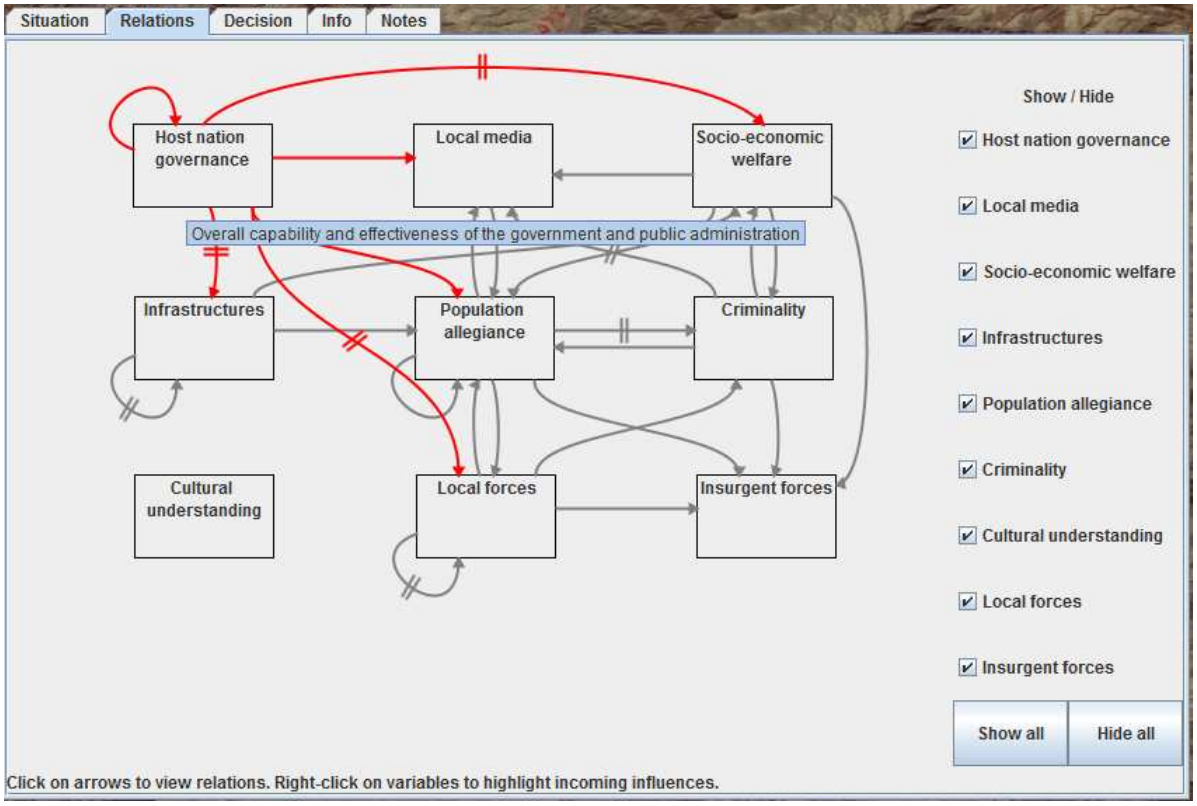
Light gray background = mediating variable, Dark gray background = agent variable

Turn # 2

Figure 1 - Situation Tab

2.2 Relations Tab

The *Relations Tab* (see Figure 2) provides a view of mutual influences on variables and allows for a user to visualize the detailed relations. Each arrow drawn in the tab represents an actual mutual influence from one variable to another. The □| |□ symbol on a line indicates that the influence will be applied on the variable after a certain delay has passed (number of turns). In this tab, the user can show or hide the outgoing relations of one variable using the checkboxes located on the right side of the tab. Moving the mouse cursor over a variable's name grays out relations that are not *outgoing* of the selected variable, i.e. those that are not direct influences. Similarly, moving the mouse cursor with the right button pressed over a variable's name grays out relations that are not *incoming* to the selected variable, i.e. those that do not have a direct influence. In order to examine a relation between two variables, the user must click on the relation arrow. If details of a mutual influence are not hidden, a chart will appear showing the associated relation (see Figure 3).

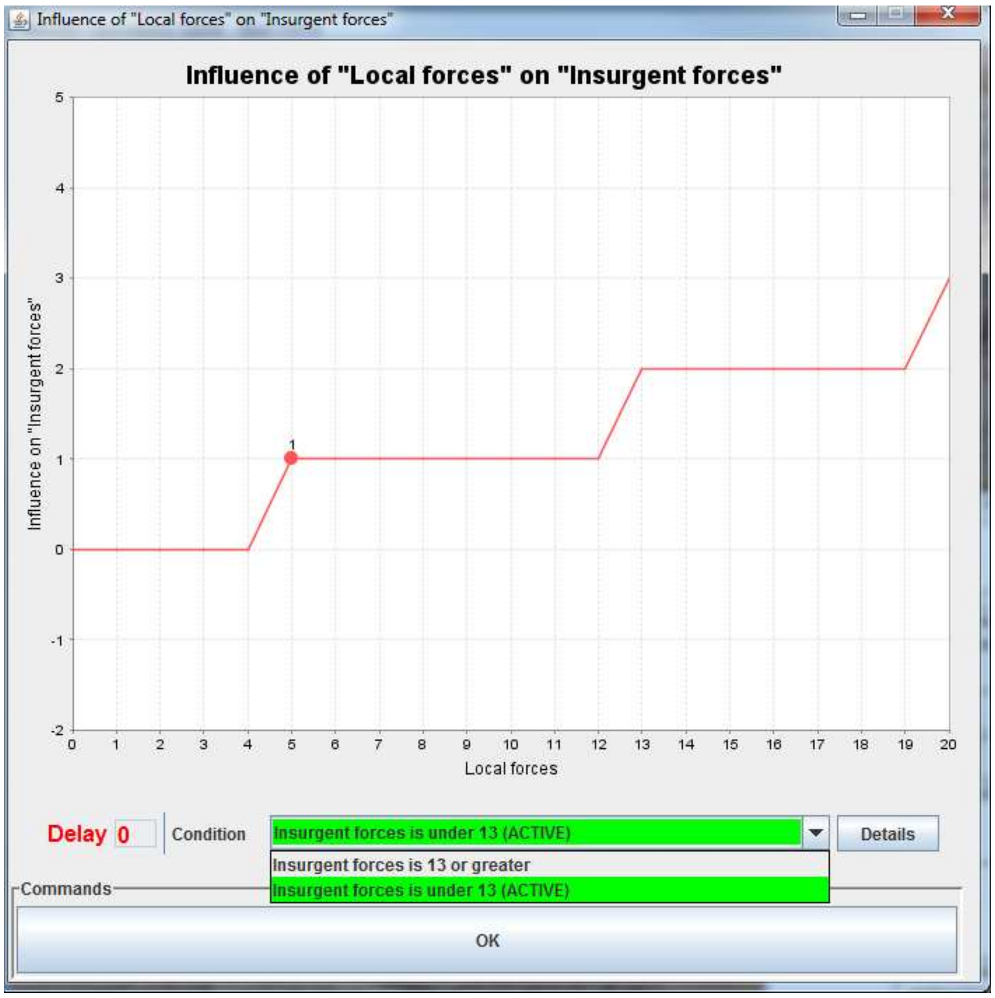


Overall capability and effectiveness of the government and public administration

Click on arrows to view relations. Right-click on variables to highlight incoming influences.

Figure 2 - Relation Tab

On the chart, a red dot indicates the current variable's value and its associated influence on the specified variable. The delay before the influence takes effect is displayed below the chart when one is specified. A "Details" button is available when the mutual influence (or conditional mutual influence) owns a "Description" parameter. In case of conditional mutual influences, condition instances are evaluated at the beginning of each turn, and condition labels are updated according to the result of this evaluation (see the bottom of Figure 3). The values of the influence relation are updated according to the user's selection in the "Condition" dropdown menu. The current active condition is highlighted in green.

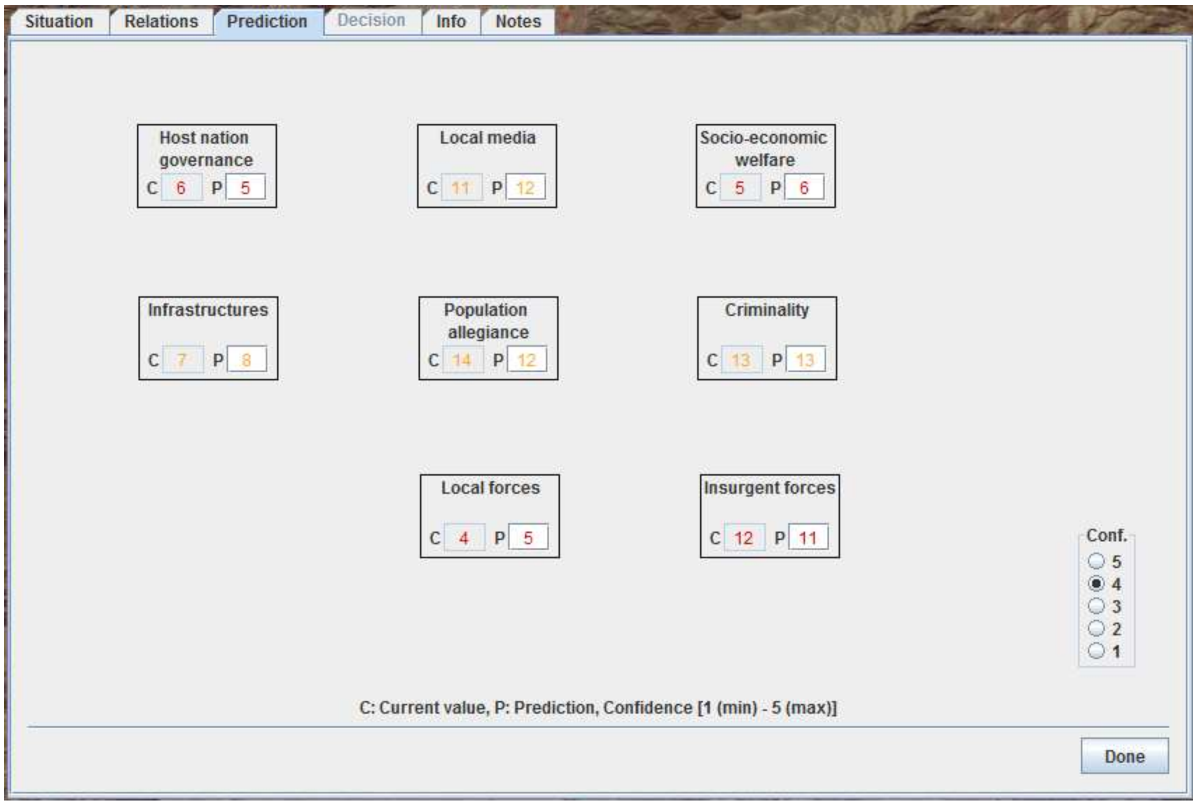


Insurgent forces is under 13 (ACTIVE)
Insurgent forces is 13 or greater
Insurgent forces is under 13 (ACTIVE)

Figure 3 - Chart of Influence

2.3 Prediction Tab

For research and cognitive testing purposes, it can be very useful to elicit participants expectations on how the situation will evolve in the next time-step. CODEM even makes it possible to create a prediction task with no interventions required. The *Prediction Tab* (see Figure 4) allows the user to make his predictions on the values of each variable for the next turn. Along with his predictions, the user can indicate the level of confidence he has in the predictions. When the prediction confidence parameter is set to Global, only one confidence level has to be entered, which is located right above the Done / Next Turn button. The current (C) value of a variable and the predicted (P) value text fields use the same color code as the coloured indicators displayed in the situation tab. The prediction tab may not be present depending on the game configuration parameters. If present, the user must complete his predictions before he can pass to the next turn. Likewise, when decisions must be made before predictions, the prediction tab will not be enabled until decisions have been made. Clicking on the “Done” (or “Next Turn” depending on the active mode) button commits predictions entered during the current turn and disables any further changes to predictions and confidence entries.



Host nation governance C 6 P 5

Local media C 11 P 12

Socio-economic welfare C 5 P 6

Infrastructures C 7 P 8

Population allegiance C 14 P 12

Criminality C 13 P 13

Local forces C 4 P 5

Insurgent forces C 12 P 11

Conf. ○ 5 [✓ 4] ○ 3 ○ 2 ○ 1

C: Current value, P: Prediction, Confidence [1 (min) - 5 (max)]

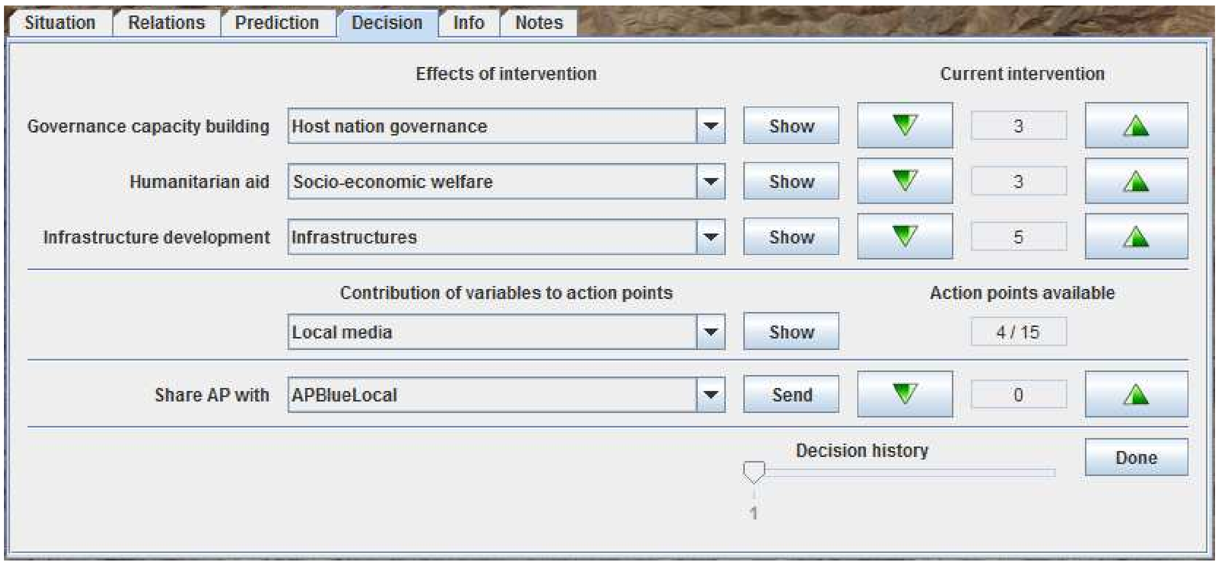
Figure 4 - Prediction Tab

2.4 Decision Tab

The *Decision Tab* (see Figure 5) allows the user to intervene in the situation using the available action points of a given turn to apply interventions. In order to decide how many points to spend on each intervention type, the user can explore effects of his intervention on different variables using the “Show” buttons. Another element to take into account is that some variables in the situation provide a number of action points for the next turn. The number of action points available depends on the variables’ value.

When several participants are collaboratively working on a mandate (same “Side” parameter), they can share action points with other participants of the same side. The participant to which action points should be sent is selected via a drop down menu, and the number of action points is adjusted with the up and down buttons. Clicking on the “Send” button sends the correct amount of action points to the selected participant. The total number of action points available and action points left are updated according to the points sent / received.

At every point in time, the user can refer to the decisions he made in the previous turns using the slider available at the bottom of the tab. When he is ready, the user spends some or all of his action points using the green arrows. He can modify his decisions until he presses the “Done” (or “Next Turn” depending on the active mode) button. A fraction of the action points that are unspent in a turn will be available in the next turn, which is configured via the “UnusedAPTransfer.”



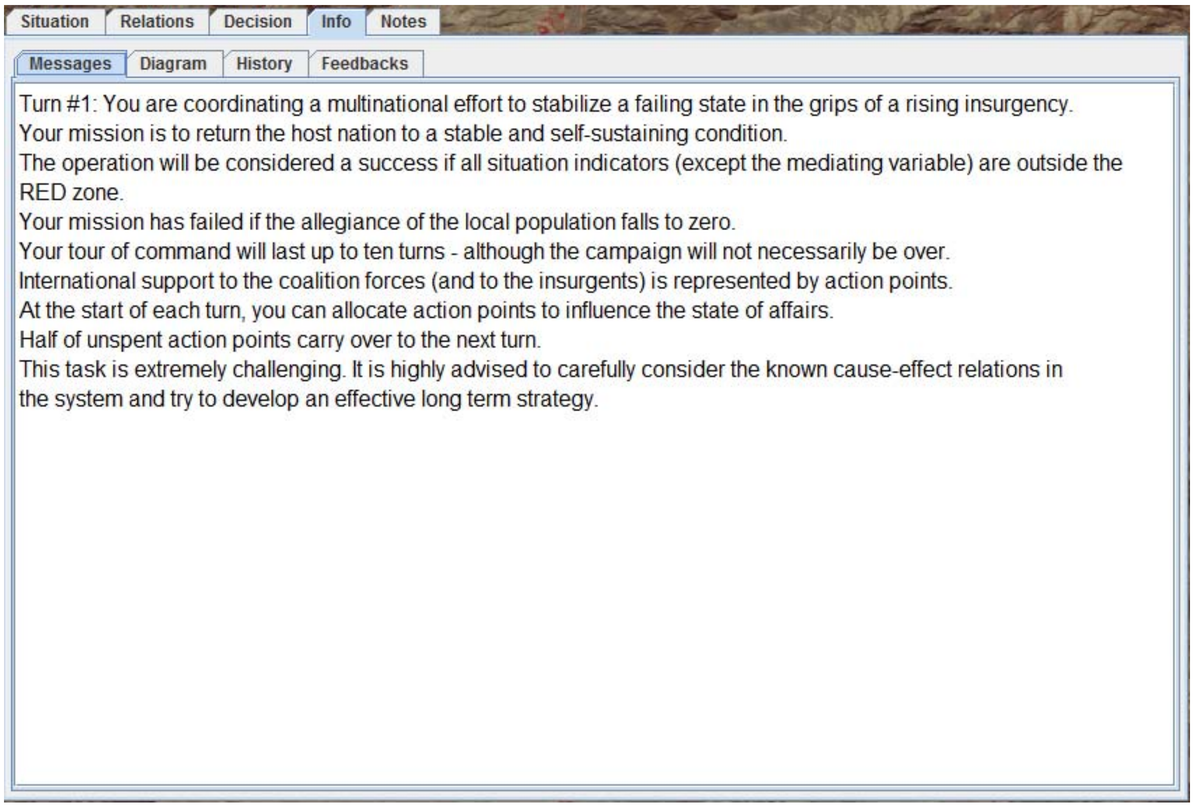
	Effects of intervention	Current intervention
Governance capacity building	Host nation governance	3
Humanitarian aid	Socio-economic welfare	3
Infrastructure development	Infrastructures	5
	Contribution of variables to action points	Action points available
	Local media	4 / 15
Share AP with	APBlueLocal	0

Decision history: 1

Figure 5 - Decision Tab

2.5 Info Tab

The *Info Tab* shows game history information in four sub-tabs. The first one, named “Messages” (see Figure 6) displays messages that are added using information text in the default category. When an information text is specified in another category, a corresponding tab is created. It should be noted that information text that was added in the current turn is displayed in black, whereas text added in previous turns are displayed in light gray in order to indicate that they are from past turns.



Turn #1: You are coordinating a multinational effort to stabilize a failing state in the grips of a rising insurgency.
Your mission is to return the host nation to a stable and self-sustaining condition.
The operation will be considered a success if all situation indicators (except the mediating variable) are outside the RED zone.
Your mission has failed if the allegiance of the local population falls to zero.
Your tour of command will last up to ten turns - although the campaign will not necessarily be over.
International support to the coalition forces (and to the insurgents) is represented by action points.
At the start of each turn, you can allocate action points to influence the state of affairs.
Half of unspent action points carry over to the next turn.
This task is extremely challenging. It is highly advised to carefully consider the known cause-effect relations in the system and try to develop an effective long term strategy.

Figure 6 - Info Tab (Messages)

The second sub-tab, named Diagram (see Figure 7), displays the evolution of the game variables. It is possible to hide some of them using the checkboxes under the chart legend.

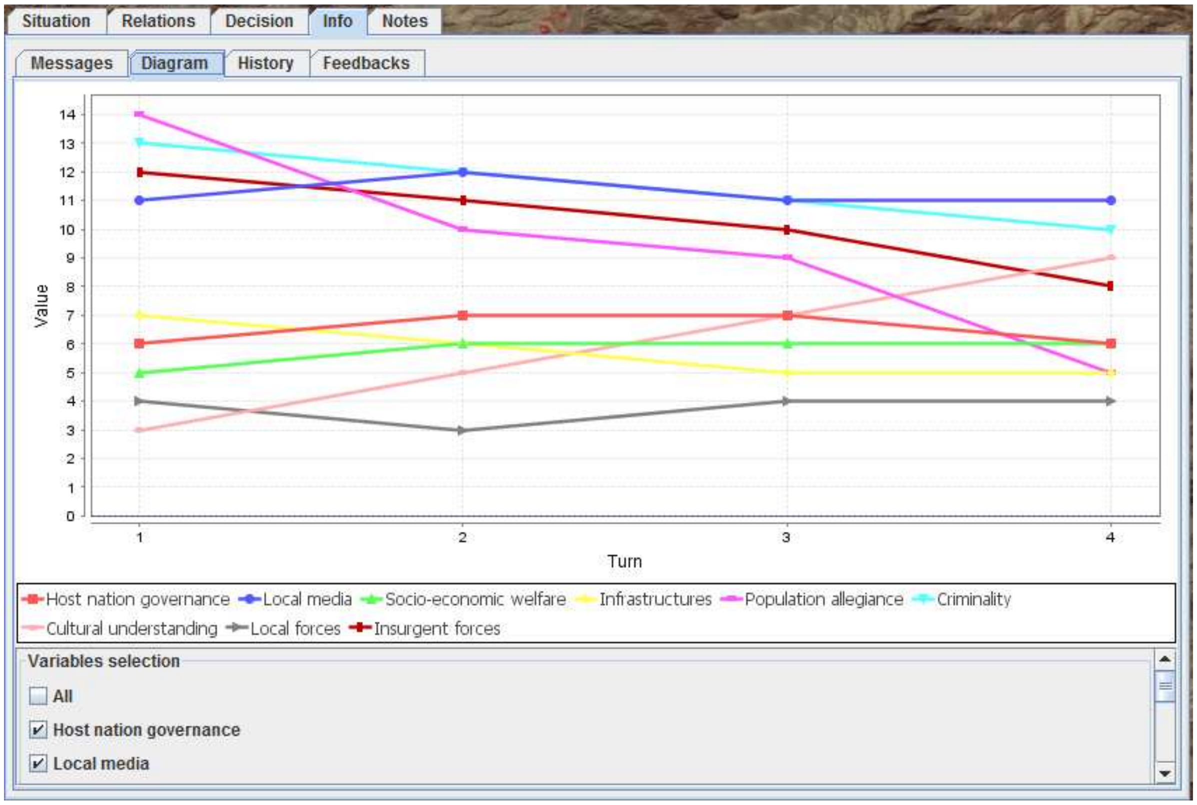
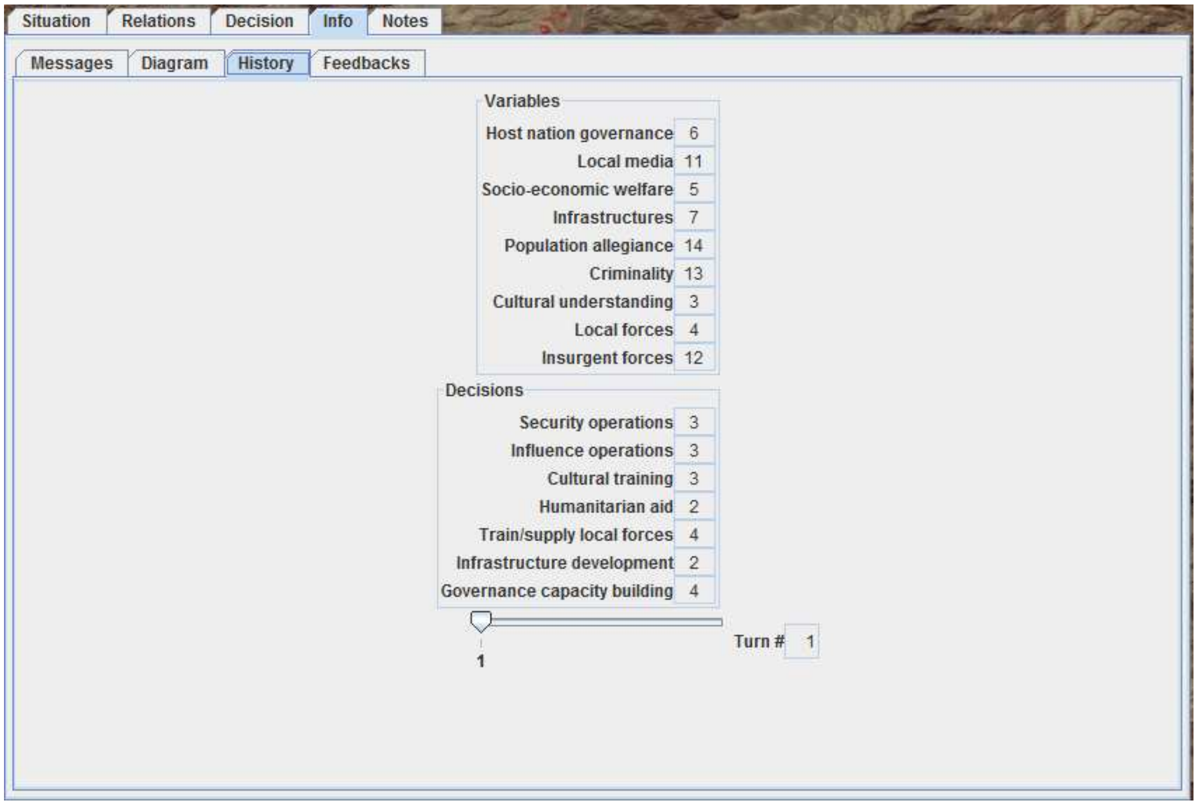


Figure 7 - Info Tab (Diagram)

The sub-tab History (see Figure 8) displays in the same view the value of all variables and the decisions made during each turn of the game. The slider should be used to go back in time.



Variables

Host nation governance	6
Local media	11
Socio-economic welfare	5
Infrastructures	7
Population allegiance	14
Criminality	13
Cultural understanding	3
Local forces	4
Insurgent forces	12

Decisions

Security operations	3
Influence operations	3
Cultural training	3
Humanitarian aid	2
Train/supply local forces	4
Infrastructure development	2
Governance capacity building	4

Figure 8 - Info Tab (History)

The last sub-tab, named Feedbacks (see Figure 9), provides history information of end of turn feedback screens (more details about those screens are provided in Section 2.7). The slider is used to go back in time, while the buttons Next and Back are used to navigate between the variable values changes and the prediction error information.

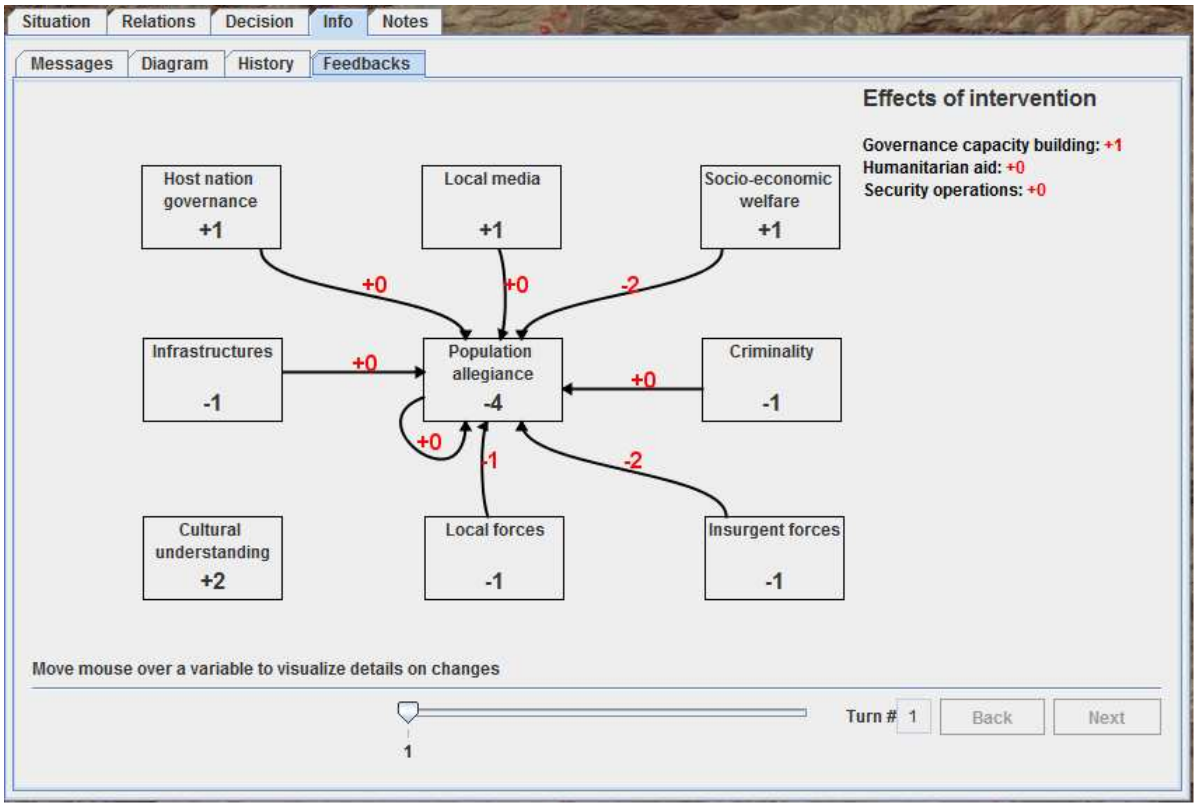
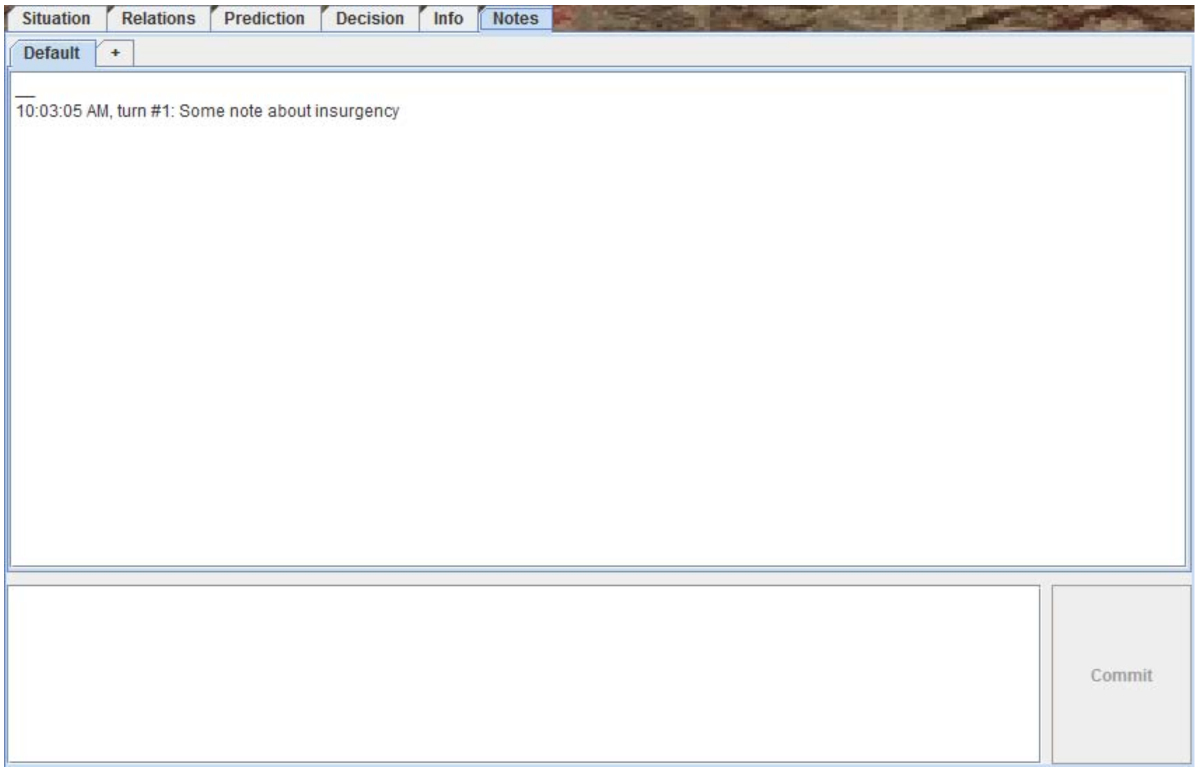


Figure 9 - Info Tab (Feedbacks)

2.6 Notes Tab

The *Notes Tab* (see Figure 10) can be used whenever needed by the user in order to take notes. If the user needs to organise his notes, tabs can be created by clicking on the “+” tab. A name can be given to a new tab at its creation. A tab can be destroyed by right-clicking on its title. Use the text field in the lower part of the tab in order to write a note and the “Commit” button in order to commit the new note to the notes history. The wall-clock time at which a note was committed as well as the current turn is used to identify the moment when a user took a particular note. If a note was not committed at the end of a turn, CODEM will automatically commit it.

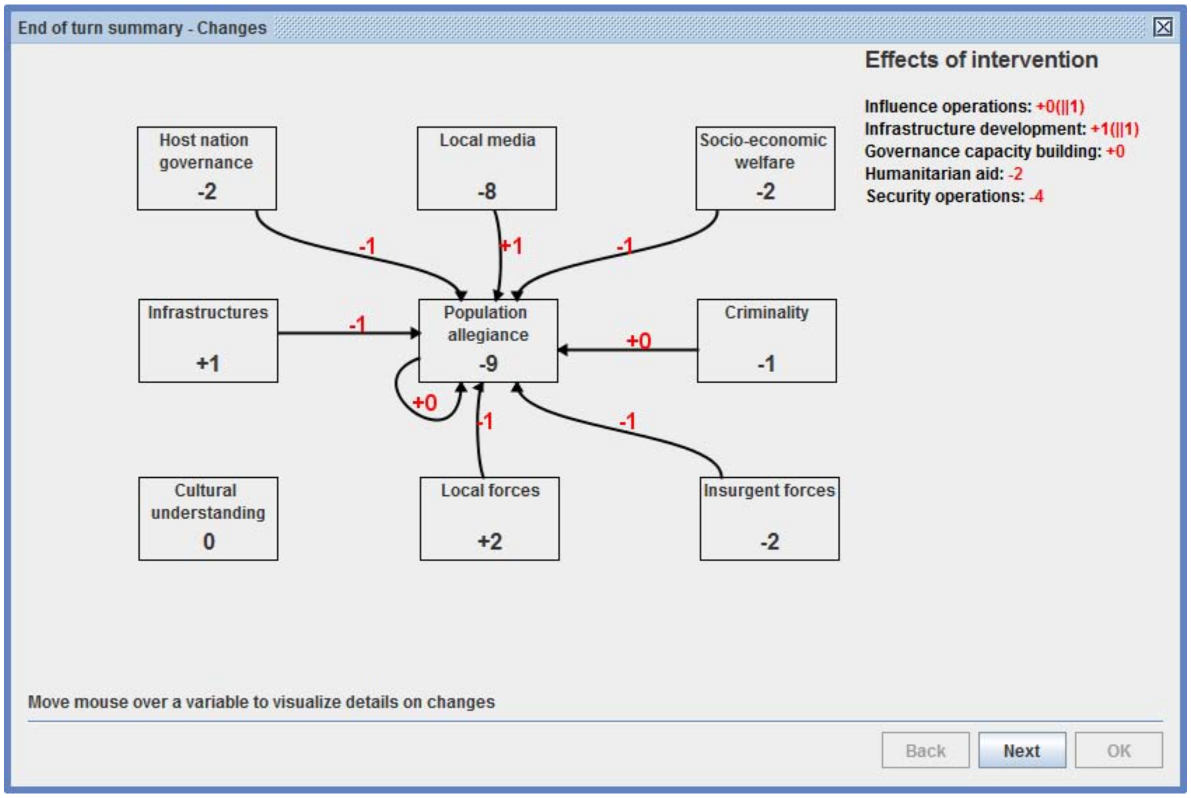


10:03:05 AM, turn #1: Some note about insurgency

Figure 10 - Notes Tab

2.7 End of Turn Feedback Screen

At the end of each turn, a feedback screen can be displayed (see Figure 11) in order to help the user understand which variables had influence on others and what have been the effects of his interventions. In order to examine this information, the user needs to move the mouse cursor over the desired variables. The actual number (preceded of �+� or �-�) specifies the absolute change for a given variable during the last turn. Incoming arrows display the contribution of corresponding variables. When the number in red is followed by �||n�, it means that the effect was delayed by �n� turns before being applied on the variable in the current turn. The effect of interventions is found on the right of the end of turn summary dialog. It displays the effect of every intervention on the variable over which the mouse cursor is placed. Right clicking on one variable displays the effect that this variable had on others using the same representation.



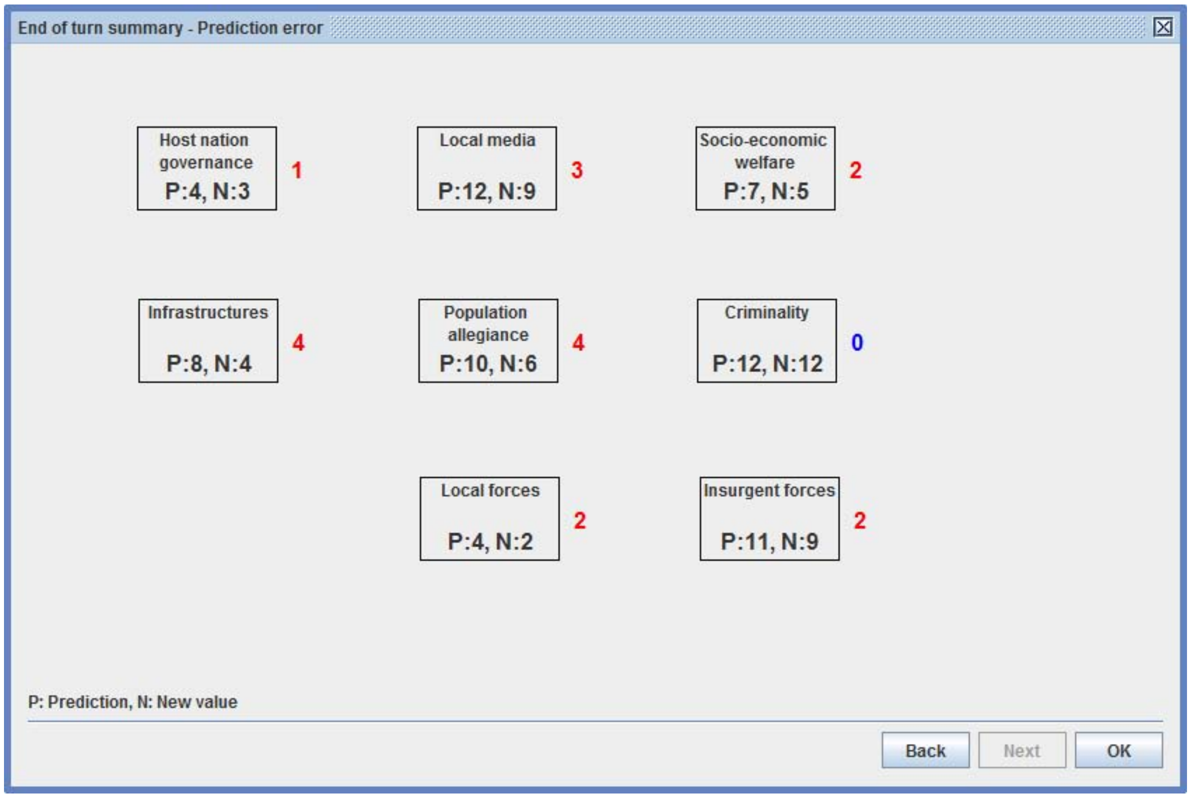
Effects of intervention

Influence operations: +0(||1)
Infrastructure development: +1(||1)
Governance capacity building: +0
Humanitarian aid: -2
Security operations: -4

Move mouse over a variable to visualize details on changes

Figure 11 - Feedback screen (End of turn Summary)

When the prediction tab is activated, another feedback screen (see Figure 12) is available. It can be shown by clicking on the "Next" button. This view displays the new variables values and the predictions made. The user can see how accurate he was in his prediction by looking at the difference between the two values indicated in red, or in blue when the prediction was accurate. The user can click on the "Back" button in order to go back to changes provenance. It should be noted that when predictions are available, the "OK" button will only be enabled when the prediction error view has been visited.



Host nation governance P:4, N:3 1

Local media P:12, N:9 3

Socio-economic welfare P:7, N:5 2

Infrastructures P:8, N:4 4

Population allegiance P:10, N:6 4

Criminality P:12, N:12 0

Local forces P:4, N:2 2

Insurgent forces P:11, N:9 2

P: Prediction, N: New value

Figure 12 - Feedback Screen (Prediction Error)

2.8 Setting the Platform to Support Multiple Participants

The experimental platform supports several participants that are making decisions in a shared situation, either collaboratively or competitively. Indeed, the same infrastructure is used with player-specific situation, relations, and intervention tabs whether participants are collaborating or competing. The first step to accomplish in order to include multiple participants in one simulation is to define a distinct action point structure and set of possible interventions for each player (see Section 0 describing the editor). Participants, each having a specific role to play in the situation, are identified by a user name and the host machine on which they will work (IP address[1] or hostname).

In order to be able to spawn user interfaces on remote machines, it is very important to start one *and only one* User Interface Dispatch Starter server on each remote machine *before* starting the main simulation application. This server actually spawns a user interface that is configured for the appropriate participant, and afterwards controls the entire data exchange between the user interface and the

[1] In order to find the IP address of a computer, run ipconfig in a command line and look for the IP address of the network adapter that is currently connected

simulator. It should be noted that the multi-participants platform uses Java Remote Method Invocation (RMI) and that it will only work in local area networks unless appropriate network ports are opened. The default port number is 2045; otherwise, it can be configured with the dispatchStarter.port system property. Note that when a port other than the default one is configured, the same system property should be set on every Java virtual machine. Also note that every participant should have access to the same version of executable files. If this is not the case, exceptions will be thrown during the objects marshalling process. For a step by step procedure on how to start CODEM for multiple participants, see Section D.1.

2.9 Log Files

One of the main objectives in using the experimental platform with human participants is to analyse the strategy they employ in their task to understand and influence a complex situation. Although the users screen is typically captured in video, which can then be analysed *a posteriori* in order to evaluate various metrics (e.g. the total time spent observing specific data, the number of times a given action is performed), such an approach can be very time consuming and prone to error. In order to support experimenters in analysing a participants actions during a session, logs are recorded to files; one for each participant that contains several metrics, and another one that contains variables values and action points allocated during a mandate. A detailed description of every log entry can be found in Appendix A.

2.10 Language Configuration

An experimenter has the ability to configure every single text label present in the user interface. For that purpose, resource bundles (simple text files), which contain text labels that should be displayed in the user interface, have to be created. System properties have to be set in order for the platform to consider the right language configuration. By default, the user interface is displayed in English. In order to use a specific resource bundle, add the following JVM arguments:

- -Dlanguage=the_language
- -Dcountry=the_country (optional)
- -Dvariant=the_variant (optional)

Or specify the above properties in the file config/CODEM.properties.

The file containing the labels can be modified via the scenario editor by choosing in the menu bar Config / Edit Labels. For more information about building a new language configuration file, follow the steps detailed in Appendix B.

3 Scenario Editor's User Guide

A scenario editor was built that allows configuring various features of the experimental platform (e.g. visual appearance, enable/disable features) as well as the complex dynamic system (e.g. variables and influences, interventions). This editor reads a configuration file that defines every single possible parameter (i.e. name, type, description) in different parts of simulation entities, and then fills a user interface with the parameter values found in a CDM scenario file that is opened and saved using the "File" menu from the menu bar. Parameters can then be modified, variables and action points created / removed, making it possible for users to build appropriate data structures that can be loaded directly into the simulator via a CDM scenario file.

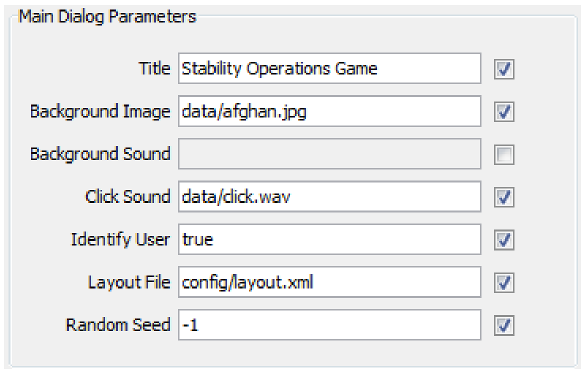
The next sections present an exhaustive list of parameters that one can set for every data structure used to configure the experimental platform via the scenario editor. A screen capture from the scenario editor shows a typical configuration of parameters. Parameters that are displayed in those screen captures originate from a counterinsurgency / stability operations scenario that was developed in order to demonstrate the capabilities of the platform.

As a general guideline for every parameter contained in the scenario editor, a label identifies the parameter's name, a text field contains the parameter's value and a checkbox, when checked, enables the actual parameter. When available (i.e., defined in the scenario editor configuration file), a *tooltip* is shown as soon as the user positions the mouse over the label or text field, which provides a description of the parameter (e.g. valid values, general instructions). Similarly, a "?" character besides the title of a group of parameters means that a tooltip will be shown when the user positions the mouse over the panel.

In order to provide greater flexibility to a scenario, it is possible to implement various conditions and statements using Javascript expressions; conditions should return a Boolean value, whereas other statements will manipulate variables or return integer values. Javascript expressions include several attributes; one for each variable, referenced by their ID, and "iteration," which is the current turn during a mandate. The reference in an expression to other attributes will result in Javascript errors. However, most common logic and arithmetic operators can be used straightforwardly.

3.1 Generic Scenario Parameters

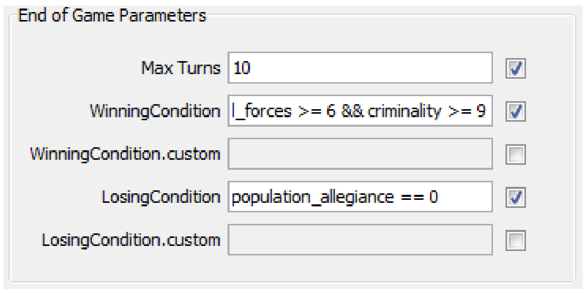
In order to edit a scenario, double-click on “CODEM_ScenarioEditor.bat” and open a CDM file via the “File / Open” menu. A scenario firstly contains several parameters that are used to configure the experimental platform’s main user interface appearance as well as features that will be enabled in it. Figure 13 shows parameters that affect the main dialog, Figure 14 displays parameters that determine conditions for the termination of a mandate, Figure 15 shows parameters that configure features available in the dialog, and Figure 16 shows parameters that determine the behaviour of the interface between turns and when the mandate is over.



Main Dialog Parameters

Title	Stability Operations Game	✓
Background Image	data/afghan.jpg	✓
Background Sound		☐
Click Sound	data/click.wav	✓
Identify User	true	✓
Layout File	config/layout.xml	✓
Random Seed	-1	✓

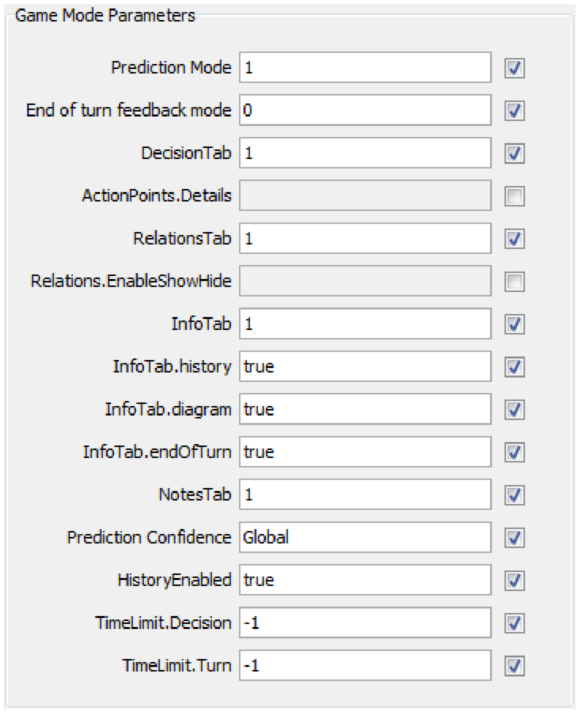
Figure 13 - Scenario parameters "Main Dialog Parameters" group



End of Game Parameters

Max Turns	10	✓
WinningCondition	l_forces >= 6 && criminality >= 9	✓
WinningCondition.custom		☐
LosingCondition	population_allegiance == 0	✓
LosingCondition.custom		☐

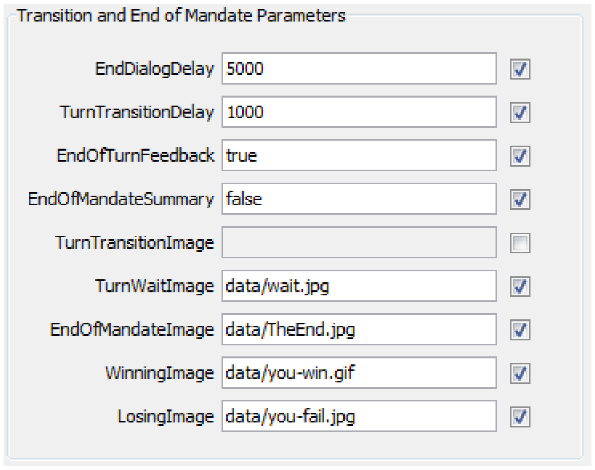
Figure 14 - Scenario parameters "End of Game Parameters" group



Game Mode Parameters

Prediction Mode	1	✓
End of turn feedback mode	0	✓
DecisionTab	1	✓
ActionPoints.Details		☐
RelationsTab	1	✓
Relations.EnableShowHide		☐
InfoTab	1	✓
InfoTab.history	true	✓
InfoTab.diagram	true	✓
InfoTab.endOfTurn	true	✓
NotesTab	1	✓
Prediction Confidence	Global	✓
HistoryEnabled	true	✓
TimeLimit.Decision	-1	✓
TimeLimit.Turn	-1	✓

Figure 15 - Scenario parameters "Game Mode Parameters" group



Transition and End of Mandate Parameters

EndDialogDelay	5000	✓
TurnTransitionDelay	1000	✓
EndOfTurnFeedback	true	✓
EndOfMandateSummary	false	✓
TurnTransitionImage		☐
TurnWaitImage	data/wait.jpg	✓
EndOfMandateImage	data/TheEnd.jpg	✓
WinningImage	data/you-win.gif	✓
LosingImage	data/you-fail.jpg	✓

Figure 16 - Scenario parameters "Transition and End of Mandate Parameters" group

Table 1 - Detailed parameters of a scenario

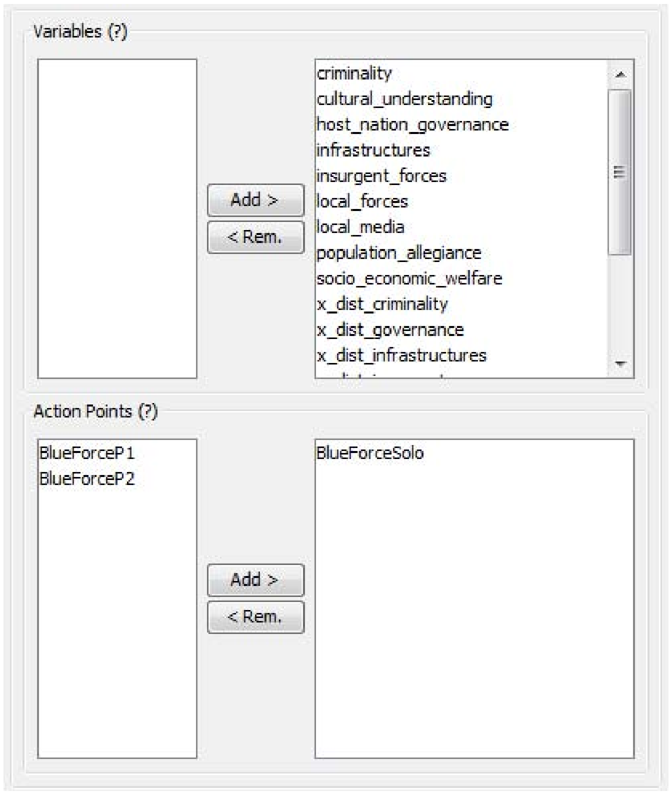
Name	Type	Valid values	Description
Title	String	Any String	Title shown in the title bar of the main user interface of the experimental platform
Background Image	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown behind the main control dialog box

Background Sound	String	Any valid sound file (wav or mp3)	File name and path relative to the root directory containing a sound that will be played in loop during the whole mandate
Click Sound	String	Any valid sound file (wav or mp3)	File name and path relative to the root directory containing a sound that will be played every time the user clicks the mouse
Identify User	Boolean	True, False	True: asks for participantID, sessionID and groupID at the beginning of a mandate; False: does not ask for user identification (default). participantID, sessionID and groupID can be set via system properties on the command line (e.g. DparticipantID=my_id)
Layout File	String	Any valid XML file name	File name and path relative to the root directory that contains the position / size of the different variable labels part of the main user interface
Random Seed	Integer	-1, ...	A seed that initializes the random number generator; -1: uses a seed that depends on time (default); other integer: fixed seed
Max Turns	Integer	-1, ...	Maximum number of turns that a mandate can last. When reached, the mandate terminates in neutral state
WinningCondition	String	Any valid Javascript expression that evaluates to a Boolean	Javascript expression that will be evaluated after each turn. When true, the mandate terminates in winning state. Variables should be referred to with their ID. Has priority over neutral and losing state
WinningCondition. custom	String	Any valid class name that implements ConditionI	When successfully loaded, an instance of this class will be evaluated after each turn. The mandate terminates in winning state when it returns true. Has priority over neutral and losing state
LosingCondition	String	Any valid Javascript expression that evaluates to a Boolean	Javascript expression that will be evaluated after each turn. When true, the mandate terminates in losing state. Variables should be referred to with their ID. Has priority over neutral state
LosingCondition. custom	String	Any valid class name that implements ConditionI	When successfully loaded, an instance of this class will be evaluated after each turn. The mandate terminates in losing state when it returns true. Has priority over neutral state
Prediction Mode	Integer	0, 1, 2	0: no prediction (default); 1: prediction first, then decision; 2: decision first, then decision
End of turn feedback mode	Integer	0, 1	0: feedback showing changes and provenance of effects (default); 1: do not show provenance of changes. Applies for both end

			of turn information in the info tab and the summary at the end of a turn
DecisionTab	Integer	0, 1	0: decision tab inactive (default); 1: decision tab active
ActionPoints.Details	String	All, Contribution, Available or None	In the decision tab - All: shows action points contribution and action points available (default); Contribution: shows only action points contribution; Available: shows only action points available; None: does not show anything
RelationsTab	Integer	0, ...	Compose an integer with activated bits, 0: inactive (default); 1: active + full features; 2: remove chart feature; 4: remove condition details feature; 8: remove condition description feature
Relations. EnableShowHide	Boolean	True, False	True: enables checkboxes for showing / hiding relations in the relations tab (default); False: disables checkboxes
InfoTab	Integer	0, 1	0: inactive, 1: active (default)
InfoTab.summary	Boolean	True, False	True: activate summary subtab in info tab; False: deactivate summary subtab (default)
InfoTab.history	Boolean	True, False	True: activate history subtab in info tab; False: deactivate history subtab (default)
InfoTab.endOfTurn	Boolean	True, False	True: activate end of turn subtab in info tab; False: deactivate end of turn subtab (default)
NotesTab	Integer	0, 1	0: inactive, 1: active (default)
Prediction Confidence	String	Yes, No, Global	Yes: need to enter a confidence value in the prediction tab for every variable; No: no need to enter a confidence value in prediction tab (default), Global: need to enter a global confidence value in the prediction tab
HistoryEnabled	Boolean	True, False	True: enables history in situation and decision tabs (default); False: does not enable history
TimeLimit.Decision	Integer	-1, ...	Number in seconds that imposes a time limit for making a decision. Countdown starts when decision tab is consulted for the first time. -1: no limit (default)
TimeLimit.Turn	Integer	-1, ...	Number in seconds that imposes a time limit for completing a turn. Countdown starts when the turn begins. -1: no limit (default)
EndDialogDelay	Integer	-1, ...	Amount of time the end dialog is shown, in milliseconds. -1 shows the dialog until it is closed by the user when no

			endOfMandateSummary. Default: 5000
TurnTransitionDelay	Integer	0, ...	Amount of time the turn transition dialog is shown, in milliseconds. Default: 3000
EndOfTurnFeedback	Boolean	True, False	True: end of turn feedback screen is shown; False: no feedback screen at the end of a turn (default)
EndOfMandate Summary	Boolean	True, False	True: end of mandate summary screen is shown; False: no screen is shown (default)
TurnTransitionImage	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown between the turns turnTransitionDelay ms.
TurnWaitImage	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown when waiting for other participants to complete their turn
EndOfMandate Image	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown when a mandate is over without winning nor losing endDialogDelay ms.
WinningImage	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown when a mandate is over in winning state endDialogDelay ms.
LosingImage	String	Any valid image file name	File name and path relative to the root directory containing an image that will be shown when a mandate is over in losing state endDialogDelay ms.

Figure 17 shows controls that allow one to define which variables and action points will be instantiated in a given scenario. On the left, variables and action points that are available but not activated for that scenario can be found. Selecting such an item and clicking the Add > button declares an instance of that type. The ID of a variable or action point is set at the time of their creation in the associated tabs. Note that IDs should not contain any special character such as spaces. The Label parameter allows specifying how variables and action points will appear in the user interface without any restriction. In order to remove an agent instance from a simulation, a user should select it in the right column and click the < Rem. button.



Variables (?)

criminality
cultural_understanding
host_nation_governance
infrastructures
insurgent_forces
local_forces
local_media
population_allegiance
socio_economic_welfare
x_dist_criminality
x_dist_governance
x_dist_infrastructures

Action Points (?)

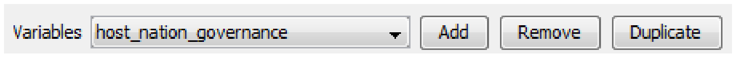
BlueForceP1
BlueForceP2

BlueForceSolo

Figure 17 - Scenario parameters "Instances" group

3.2 Description of Variables

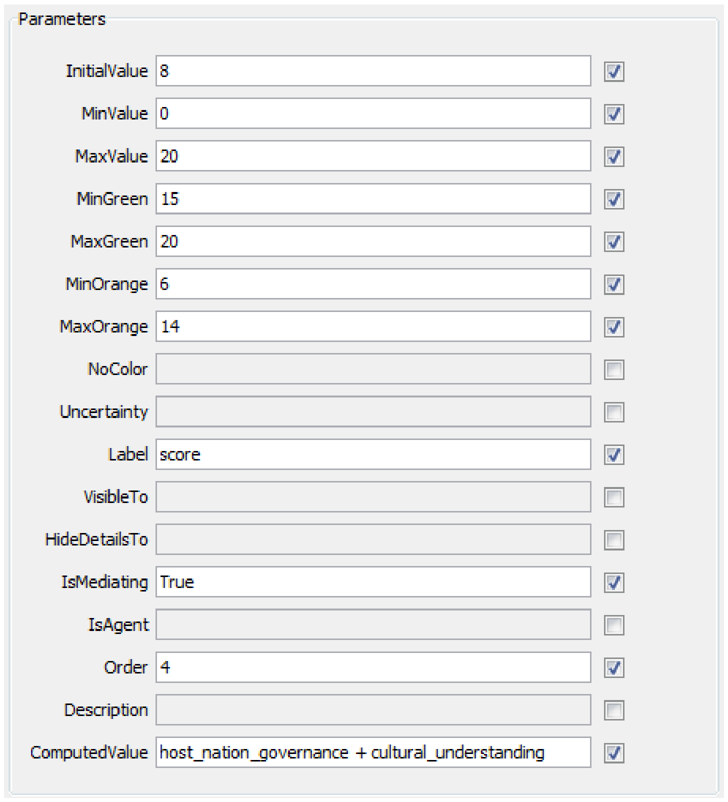
A set of variables and their mutual influences define a complex situation that is updated after every turn. The properties of variables should be defined with the scenario editor. Figure 18 shows controls that are used to select, add and remove variables from the set of defined variables. Clicking the “Add” button asks the user to name a variable. Removing a variable deletes the corresponding data structure. Selecting a variable updates the value and enabled state of every single parameter associated to that variable. It is possible to duplicate the data structure associated to a variable via the “Duplicate” button. The user is then asked to name the new variable.



Variables host_nation_governance

Figure 18 - Variable addition and removal controls

Simple parameters are associated to every variable (Figure 19). Those include minimum and maximum values, the configuration of target values and the initial value of a variable. In particular, parameters that specify the range of values associated to “green,” “orange,” and “red” categories are found in simple parameters (Figure 20). More details on all the parameters can be found in Table 2.



Parameters

InitialValue	8	✓
MinValue	0	✓
MaxValue	20	✓
MinGreen	15	✓
MaxGreen	20	✓
MinOrange	6	✓
MaxOrange	14	✓
NoColor		☐
Uncertainty		☐
Label	score	✓
VisibleTo		☐
HideDetailsTo		☐
IsMediating	True	✓
IsAgent		☐
Order	4	✓
Description		☐
ComputedValue	host_nation_governance + cultural_understanding	✓

Figure 19 - Simple variables parameters

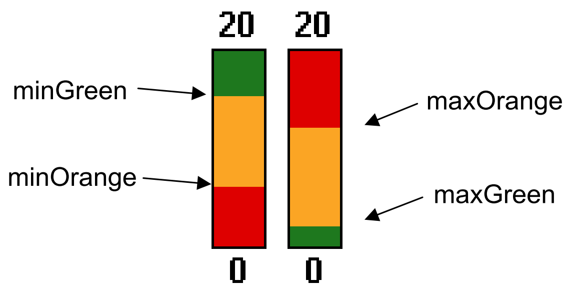


Figure 20 - Details of the color categories for variables values

Table 2 - Details of simple variables parameters

Name	Type	Valid values	Description
InitialValue	Integer	MinValue, MaxValue	Value that should be found within the range MinValue and MaxValue that is the initial value of that variable
MinValue	Integer	Any integer lower than MaxValue	The minimum value of that variable
MaxValue	Integer	Any integer larger than MinValue	The maximum value of that variable

MinGreen	Integer	MinValue, MaxValue	The minimum value that defines the green range
MaxGreen	Integer	MinGreen, MaxValue	The maximum value that defines the green range
MinOrange	Integer	MinValue, MaxValue	The minimum value that defines the orange range
MaxOrange	Integer	MinOrange, MaxValue	The maximum value that defines the orange range
NoColor	Boolean	True, False	True: remove green, orange and red ranges from variables representations and fills the range with a blue color; False: fills the range with specified green, orange and red colors (default)
Uncertainty	Integer	0, ...	When not 0, the real value of a variable is found inside a +/- uncertainty range around the displayed value. Only affects values displayed in the interface, not in the simulation. Default: 0
Label	String	Any valid String	The label that identifies a variable in the user interface
VisibleTo	List<String>[2]	Any valid list of Strings	[all]: visible to all users (default); [none]: visible to nobody; [user1, user2]: list of participants for which this variable will be visible in the interface
HideDetailsTo	List<String>	Any valid list of Strings	[none]: hide details to nobody (default); [all]: hide details to all users; [user1, user2]: list of participants for which details of this variable will be hidden (value not available) in the interface
IsMediating	Boolean	True, False	True: when a variable is considered as a mediating variable, representing it differently in the situation tab, and removing it from prediction tab; False: variable is not mediating (default)
IsAgent	Boolean	True, False	True: when a variable is considered as an “agent” variable, representing it differently in the situation tab; False: variable is not “agent” (default)
Order	Integer	Any integer	The number of appearance of that variable when predefined layout is unavailable. Smaller values are sorted first
Description	String	Any valid String	A description of the variable that is shown as a tooltip of variable labels in situation, relation and prediction tabs[3]
Computed Value	String	Any valid Javascript expression that evaluates to an	An expression that overrides the value set by mutual influences. It is evaluated at the beginning of a turn and takes into account the values of the

[2] A general guideline for lists is to put elements around bracket characters (‘[’ and ‘]’), separated by a comma (‘,’)
[3] For descriptions that are longer than one line of text, new lines can be created by using the HTML notation. For that purpose, begin the description with “<html>”, create a new line by inserting “<p/>” and end your description using “</html>” (e.g. <html>First line <p/> Second line </html>). The same comment applies to every field that acts as a description object in “Variables” and “Action Points”

	Integer	previous turn. Computed values are processed in the order set by the “Order” parameter

3.2.1 Mutual Influences

Variables influence each other via *mutual influence* relations. A mutual influence is a relation that associates a relative change applied to a variable and the current value of a variable. As an example, Figure 21 shows influence values. For instance, when the domain value is “10,” the influence is “0.” However, when the domain value is “18,” the influence is “2.” The experimenter can configure the opacity (how much details are shown) of a mutual influence in the main user interface in several ways. Firstly, it is possible to hide completely a relation from the user. Therefore, the user will not be aware that a relation between two variables exists. Secondly, it is possible to disable revealing details of the actual relation to a user. Therefore, the user is aware of the presence of a relation but cannot access the actual influence values. Thirdly, it is possible to display “dummy” relations that replace “real” relations. Therefore, the user is aware of a relation that can exist or not in the simulation. It is the experimenter’s role to specify the configuration of each relation, and to notify the participant accordingly via the “description” parameter or information texts.

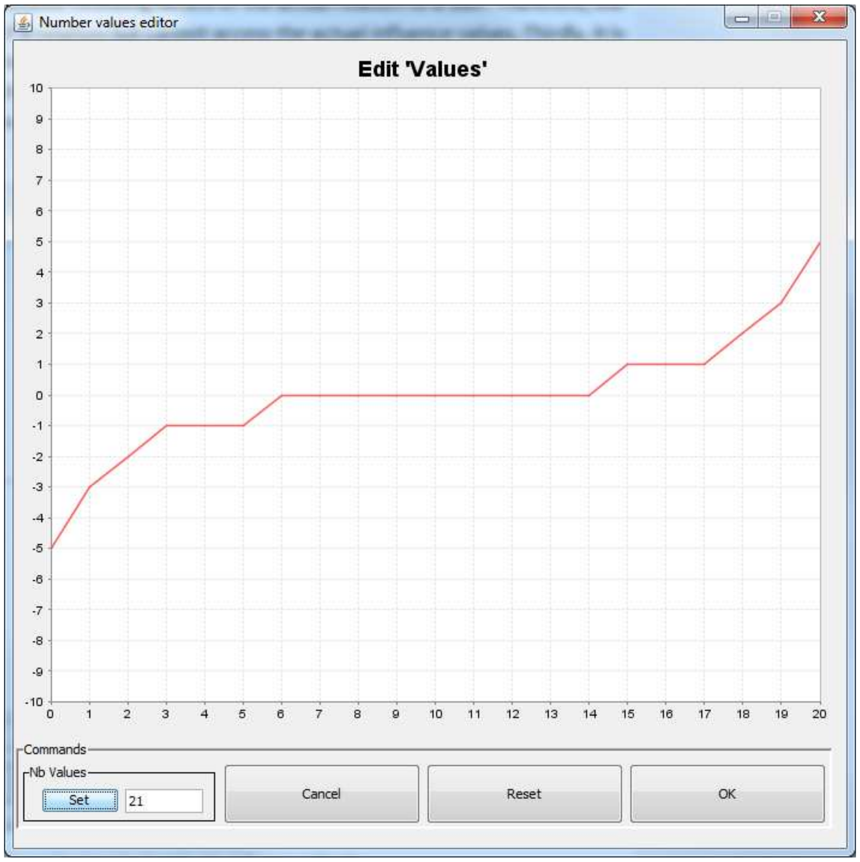
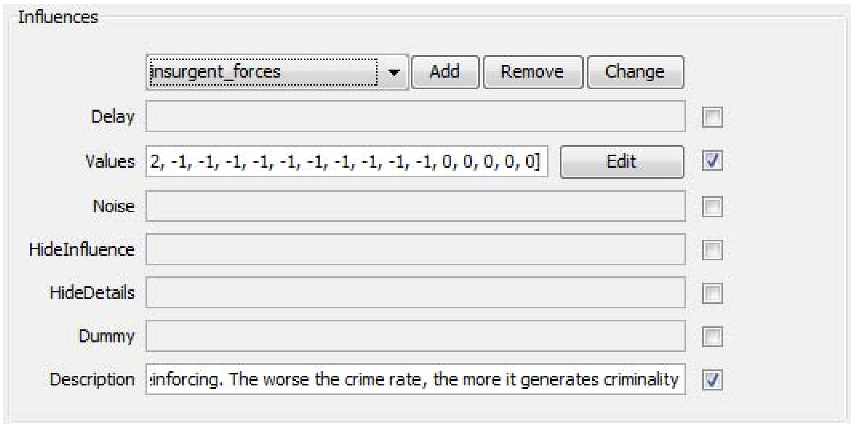


Figure 21 - Sample chart for the influence editor

In order to set mutual influences with the scenario editor, one should look into the Variables tab in the Influences section (Figure 22). Clicking on the Add button will ask the user to specify the variable for which the mutual influence should be created. The actual variable should be selected from a combo box that contains every instantiated variable (Figure 17). Clicking the Remove button will delete the selected mutual influence. Clicking the Change button asks the user to choose a replacement variable for the currently selected influence. More details on every parameter can be found in Table 3. Note that clicking on the Edit button associated to the Values parameter will display a chart similar to the one shown in Figure 21 in which values can be edited by the user. It is thus easier for a user to specify a relation by dragging the cursor on a chart rather than modifying values by hand. The Set button is used to set the number of values present in the domain, while clicking Cancel cancels any change, Reset resets values to the state they were when the dialog was opened and OK commits values to the scenario editors text field.



	insurgent_forces	
Delay		☐
Values	2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0]	✓
Noise		☐
HideInfluence		☐
HideDetails		☐
Dummy		☐
Description	:inforcing. The worse the crime rate, the more it generates criminality	✓

Figure 22 - Variables' mutual influences parameters

Table 3 - Details of "Influences" parameters

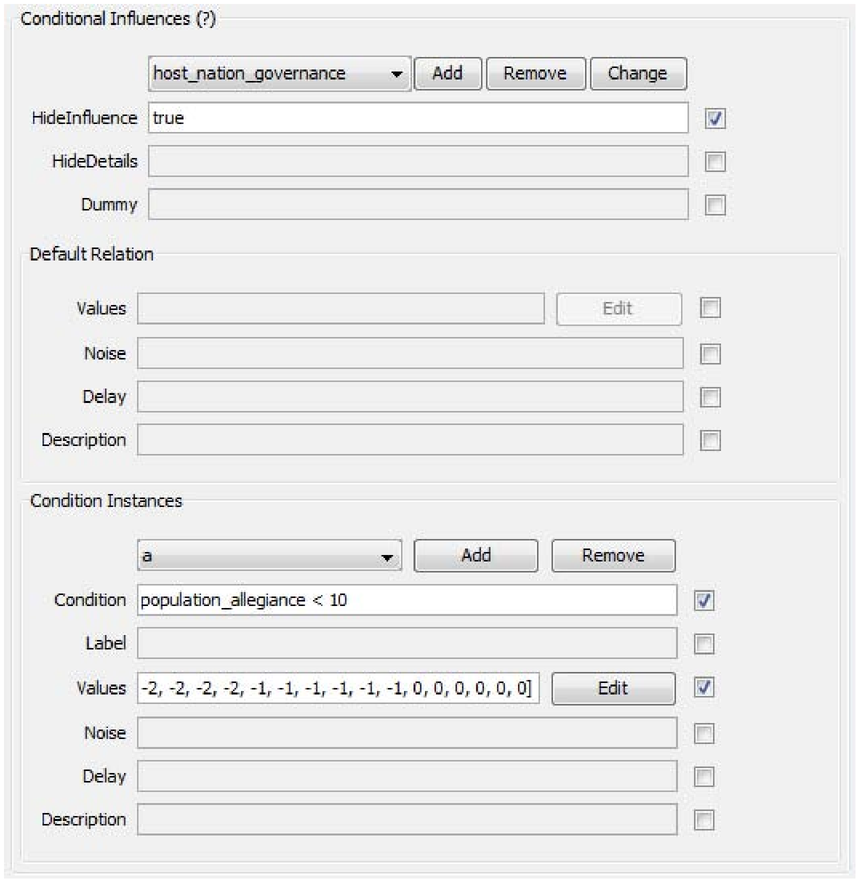
Name	Type	Valid values	Description
Delay	Integer	0, ...	A delay introduced in the application of the mutual influence. Shown in the chart dialog
Values	List<Integer>	Any list of integers	The actual influence on the specified variable for each value that the current variable can take. The size of the list should be maxValue-minValue+1, except for dummy relations that can be of any size
Noise	Integer	0, ...	Random noise added to the influence value. Default: 0
HideInfluence	Boolean	True, False	True: hides the relation from the user interface; False: shows the relation in the user interface (default)
HideDetails	Boolean	True, False	True: disables clicking the relation in the user interface; False: enables clicking the relation in the

			user interface (default)
Dummy	Boolean	True, False	True: displays a relation in the user interface but does not consider it in the simulation; False: the relation is considered in the simulation (default). Dummy has priority over “real” relations for being displayed in the user interface
Description	String	Any valid String	A description of the current relation that will be available in the chart dialog with the “Details” button

3.2.2 Conditional Mutual Influences

In order for the mutual influence of a variable to change according to the situation’s state, it is possible to specify “Conditional Influences” (Figure 23). For that purpose, a *default* relation can be specified (“Default Relation” section) as well as one or several condition instances. Clicking the “Add” button in condition instances will create a new data structure that stores condition instances and will automatically assign an ID to it. The latter data structure contains a “Condition” parameter that is in fact a Javascript statement[4], which is evaluated to either “true” or “false.” Table 4 shows a summary of most frequently used Javascript operators. The first condition instance that is evaluated to true will be considered. There is no specific order specification, so the scenario designer should make sure that conditions are mutually exclusive, i.e. that only one is evaluated to true at the same time. When all conditions are evaluated to false, the default relation is used. The “Label” parameter is used to replace the condition statement when a user requests details about a conditional influence. It therefore simplifies the statement that identifies a condition, but is optional. By default, the actual condition is displayed. Excluding the “Condition” and “Label” parameters, the others have the same signification as those contained in “Influences.” Adding and removing conditional mutual influences uses the same principle as mutual influences. It should also be noted that conditional mutual influences have precedence over mutual influences, which means that when conditional and normal influences affect a given variable, the conditional influence has priority when calculations are performed in the simulator.

[4] For more details about the Javascript syntax, see https://developer.mozilla.org/en/Rhino_documentation. External Java classes are limited to “java.lang.Math”. Note that there is a potential security breach here because arbitrary code can be executed (http://codeutopia.net/blog/2009/01/02/sandboxing-rhino-in-java/)



	host_nation_governance	
HideInfluence	true	✓
HideDetails		☐
Dummy		☐

Default Relation

Values		☐
Noise		☐
Delay		☐
Description		☐

Condition Instances

	a	
Condition	population_allegiance < 10	✓
Label		☐
Values	-2, -2, -2, -2, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0]	✓
Noise		☐
Delay		☐
Description		☐

Figure 23 - Variables' conditional mutual influences parameters

Table 4 - Summary of Javascript operators

Operator	Description
==	Equality operator
!=	Inequality operator
||	Boolean or operator
&&	Boolean and operator
<	Boolean smaller than operator
<=	Boolean smaller than or equal operator
>	Boolean larger than operator
>=	Boolean larger than or equal operator
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Modulo operator
=	Assignment operator
+=	Combined addition and assignment operator
-=	Combined subtraction and assignment operator

*=	Combined multiplication and assignment operator
/=	Combined division and assignment operator
if(cond.) {code;} else if{code;} else {code;}	Conditional statement code returns a value (no return statement needed) or manipulates variables

3.3 Description of Action Points

Action points allow a user to have an effect on how the situation will evolve in time via interventions on the situation. At the beginning of each turn of a mandate, a number of action points is allocated to each participant. This number typically depends on preset parameters and on the situations state. Action points obey to the same principle as mutual influences except that the influence on a variable is not a function of a variables value, but rather a function of the number of action points that are allocated on a given intervention. A fraction of the action points that are not allocated during a turn will be available in the following turn, which can be configured via the UnusedAPTransfer parameter. Adding and removing action points in a scenario is performed using controls that are shown in Figure 24. A new or duplicated action point data structure has to be named with a unique ID for future reference. More details on parameters available in the configuration of action points can be found in Table 5.

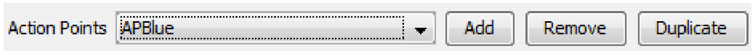
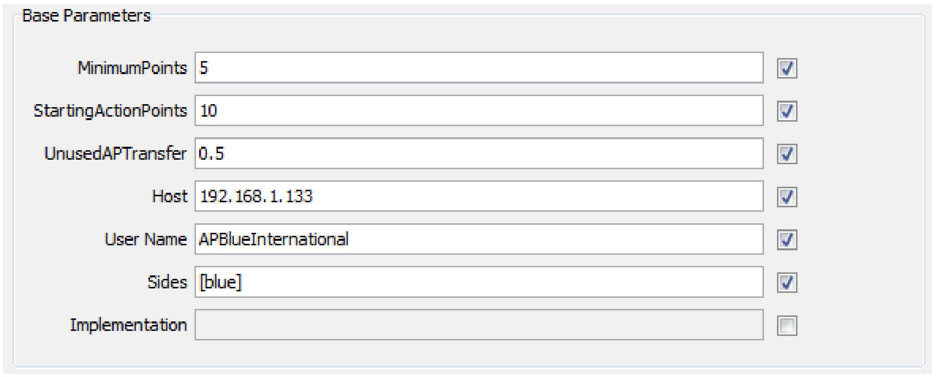


Figure 24 Action points addition and removal controls



Base Parameters

MinimumPoints	5	✓
StartingActionPoints	10	✓
UnusedAPTransfer	0.5	✓
Host	192.168.1.133	✓
User Name	APBlueInternational	✓
Sides	[blue]	✓
Implementation		☐

Figure 25 - Action points base parameters

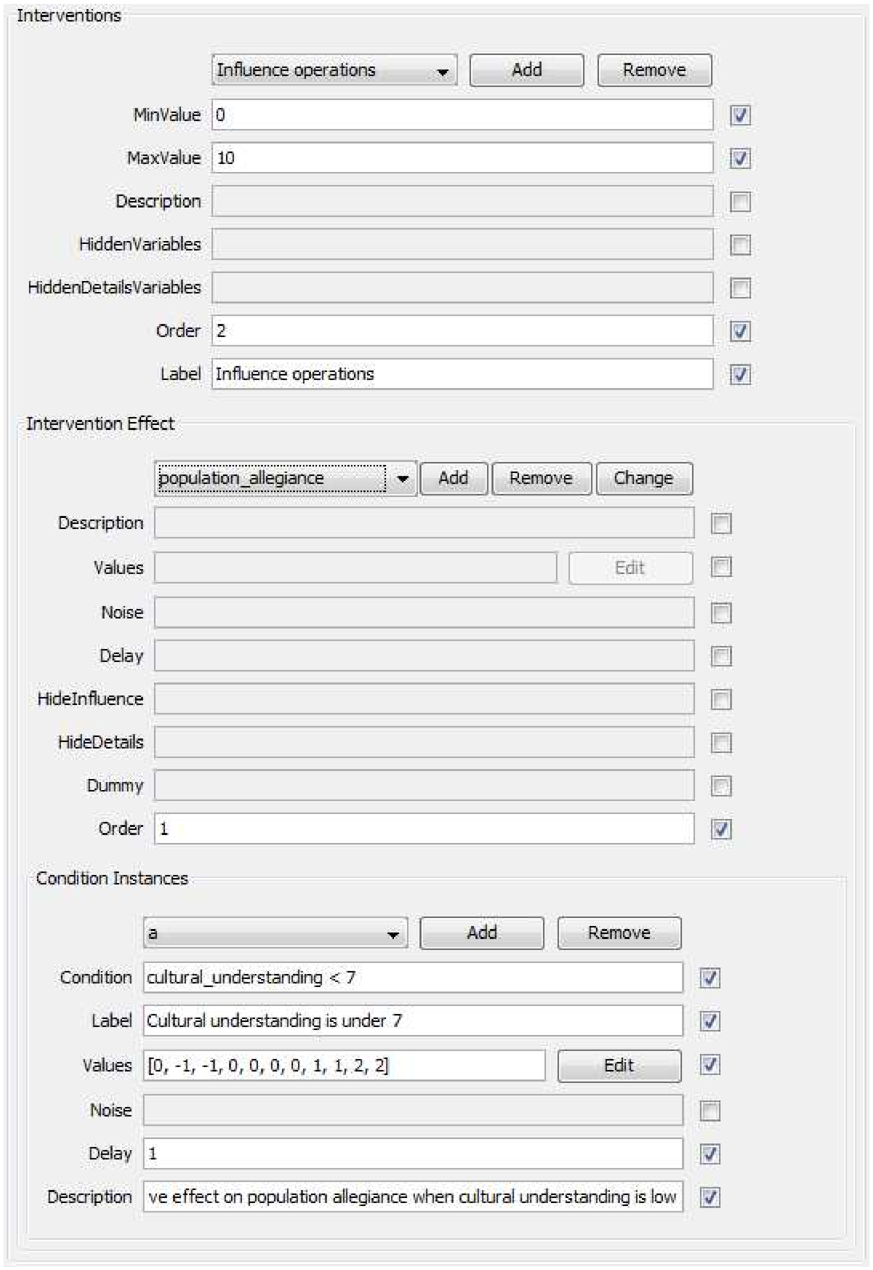
Table 5 - Details of parameters associated to Action Points

Name	Type	Valid values	Description
MinimumPoints	Integer	0, ...	The minimal number of action points available at the beginning of a turn. Contribution of variables and unused action points are added

			to this number to calculate the total number of available action points in a given turn
StartingAction Points	Integer	-MinimumPoints, …	The initial number of additional action points. Replaces variables contribution in the first turn
UnusedAPTransfer	Float	0.0-1.0	Proportion of unused action points transferred to the next turn
Host	String	Any valid hostname or IP address	The host on which the specified participant will be performing his work. Use localhost in order to spawn the main user interface in the simulators process. Note that only one user is supported per computer
User Name	String	Any valid String	A name identifying the user that will be part of the main dialogs title and the users log file name
Sides	List<String>	Any valid list of Strings	List of sides that this participant is part of. Participants that are part of the same sides can share action points. Default: [] (no sides)
Implementation	String	Any valid class name that implements appropriate interfaces	Specifies the class that implements an actual participant. Local and remote participants are distinguished, i.e. local-only participants should implement UIDispatchAdapterI and remote participants should implement UIRemoteDispatchAdapterI. Default: default uses the default implementation that displays the standard user interface

3.3.1 Interventions

Interventions that a user can perform on the situation are also specified in the Action Points tab of the scenario editor. Parameters include the minimum and maximum values as well as the relations defining how interventions influence variables of a situation. Figure 26 shows the details of the interface associated to intervention parameters in the scenario editor, while Table 6 and Table 7 provide additional details regarding valid values and a description of every single parameter and intervention effects respectively. Note that interventions can either influence variables through a default relation, or through relations that are activated under certain conditions that depend on the variables values (Condition Instances).


Interventions

Influence operations | Add | Remove

MinValue	0	✓
MaxValue	10	✓
Description		
HiddenVariables		
HiddenDetailsVariables		
Order	2	✓
Label	Influence operations	✓

Intervention Effect

population_allegiance | Add | Remove | Change

Description		
Values		
Noise		
Delay		
HideInfluence		
HideDetails		
Dummy		
Order	1	✓

Condition Instances

a | Add | Remove

Condition	cultural_understanding < 7	✓
Label	Cultural understanding is under 7	✓
Values	[0, -1, -1, 0, 0, 0, 0, 1, 1, 2, 2]	✓
Noise		
Delay	1	✓
Description	ve effect on population allegiance when cultural understanding is low	✓

Figure 26 - Interventions parameters

Table 6 - Details of basic Interventions parameters

Name	Type	Valid values	Description
MinValue	Integer	Any valid Integer lower than MaxValue	The minimum value for that intervention. Note that negative minimum values are special because the neutral intervention happens when the value is zero. Therefore, negative interventions are tagged accordingly (+/- sign) in the user interface
MaxValue	Integer	Any valid Integer	The maximum value for that intervention

		larger than MinValue	
Description	String	Any valid String	A description of that intervention, which is shown as the tooltip of the intervention's label in the main user interface
Hidden Variables	List<String>	Any valid list of Strings containing variable IDs	Intervention effects of specified variables are systematically hidden (i.e., not shown to participants). Empty list by default
HiddenDetails Variables	List<String>	Any valid list of Strings containing variable IDs	Intervention effects relation details of specified variables are systematically hidden (i.e., the presence of relations are shown, but not the specific effects). Empty list by default
Order	Integer	Any valid Integer	Intervention effects are ordered in the user interface according to the order specified. Lower values are sorted first
Label	String	Any valid String	An intervention effect is identified with this label in the user interface

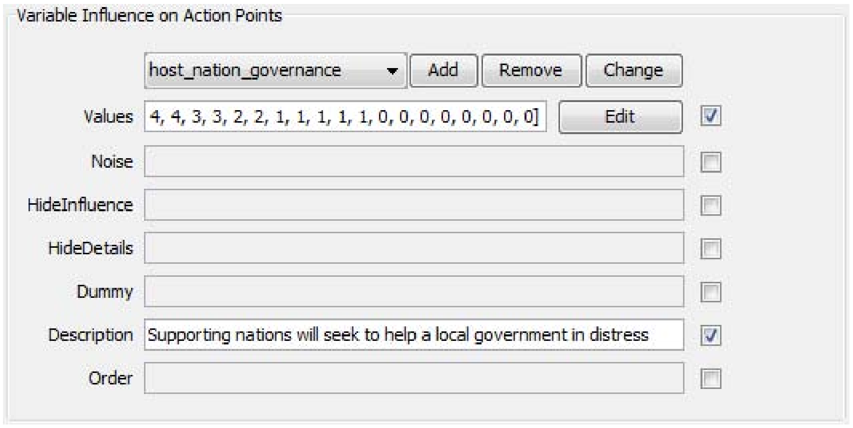
Table 7 - Details of parameters associated to Intervention Effects

Name	Type	Valid values	Description
Description	String	Any valid String	A description of this intervention effect that is shown when the user clicks the "Details" button in the relation's chart
Values	List<Integer>	Any valid list of Integers	Specifies the intervention's influence on the current variable. The size of that list should be "MaxValue – MinValue + 1", except for dummy interventions that can be of any size
Noise	Integer	0, ...	Random noise added to the intervention value. Default: 0
Delay	Integer	0, ...	The delay before the influence is actually applied on the variable's value
HideInfluence	Boolean	True, False	True: hides the intervention effect from the list displayed to the user; False: adds the intervention effect in the list (default)
HideDetails	Boolean	True, False	True: disables showing the relation chart of variable influence as a function of the intervention value; False: enables showing the relation (default)
Dummy	Boolean	True, False	True: the intervention effect is displayed to the user but not taken into account in the simulation; False: this is a real intervention effect (default)
Order	Integer	Any valid Integer	The order in which intervention effects will be presented in the dropdown menu. Lower values are sorted first
Condition	String	Any Javascript statement that returns a Boolean	The condition under which a conditional intervention effect is considered rather than the default values relation

Label	String	Any valid String	A label that will be displayed instead of the actual condition when a user requests for details about a condition instance. Default: the actual condition
Values	List<Integer>	Any valid list of Integers	Same as default values but for a conditional intervention effect. Default values relation is considered when all conditions return false

3.3.2 Variable Influence on Action Points

In a typical scenario, the number of action points to be dispatched by the user will change according to the state of the situation. The contribution of each variable to this number is specified in the Action Points tab of the scenario editor. Parameters available are shown in Figure 27, and additional details can be found in Table 8.



Values	4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]	✓
Noise		☐
HideInfluence		☐
HideDetails		☐
Dummy		☐
Description	Supporting nations will seek to help a local government in distress	✓
Order		☐

Figure 27 Variable influence on action points

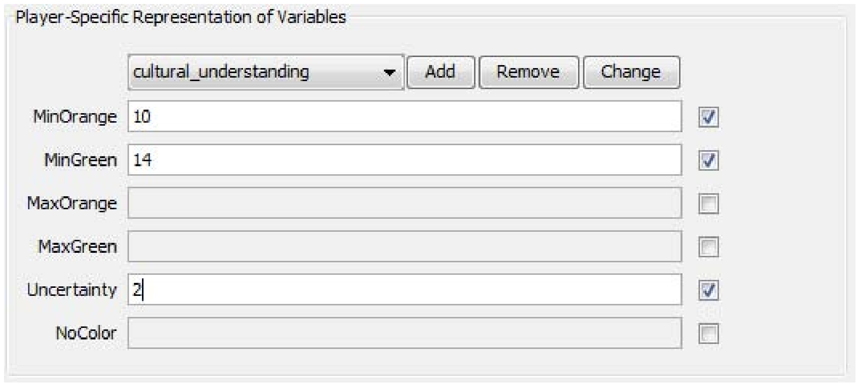
Table 8 - Details of variable influence on action points parameters

Name	Type	Valid values	Description
Values	List<Integer>	Any valid list of Integers	List that specifies the contribution of a given variable to the number of action points as a function of its value. The size of that list should be variable.MaxValue variable.MinValue + 1, except for dummy contributions that can be of any size
Noise	Integer	0, ...	Random noise added to the influence value. Default: 0
HideInfluence	Boolean	True, False	True: hides the action point provenance from the list displayed to the user; False: adds the action point provenance to the list (default)

HideDetails	Boolean	True, False	True: disables showing the relation of action points contribution as a function of the variable�s value; False: enables showing the relation (default)
Dummy	Boolean	True, False	True: the action point provenance is simply displayed to the user and not taken into account in the calculations; False: this is a real action points provenance (default)
Description	String	Any valid String	A description of that action points provenance that is shown when the user clicks the �Details� button in the relation chart
Order	Integer	Any valid Integer	Action points�provenances are organized in the user interface according to the order specified. Lower values are sorted first

3.3.3 Player-Specific Representation of Variables

In a scenario involving multiple participants, variables may not have the same meaning for every participant. For that purpose, a player-specific representation (green, orange and red ranges) can be defined for every variable, which is different for every participant (in the action points tab). Similarly, the uncertainty that is displayed around a variable can be specified for every user. Specific representations values are defined in the �Action points� tab of the scenario editor. Figure 28 shows a typical configuration of a player-specific representation in the scenario editor. Clicking on the �Add� button asks the user for which variable he wants to define a new player-specific representation of variable. Clicking on the �Remove� button removes the selected specific representation. The signification of parameters is the same as those established in Table 2.



	cultural_understanding	
MinOrange	10	✓
MinGreen	14	✓
MaxOrange		☐
MaxGreen		☐
Uncertainty	2	✓
NoColor		☐

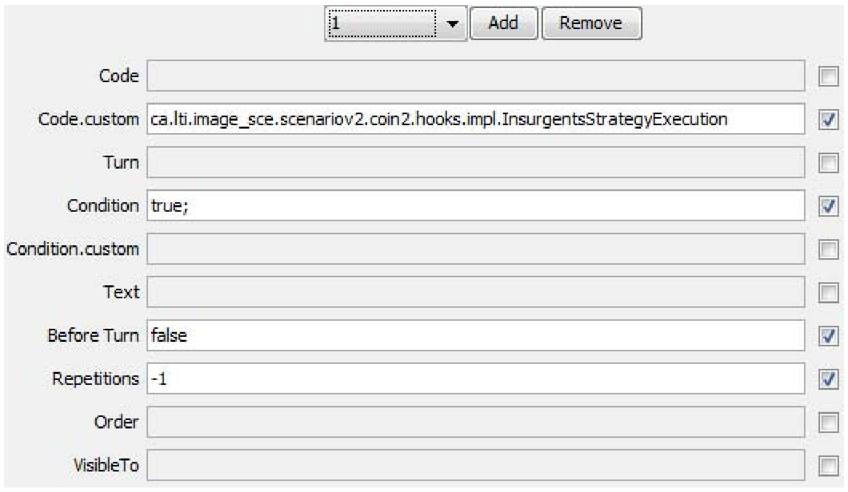
Figure 28 � Player-specific representation of variables

3.4 Description of Custom Events

Custom events allow a scenario developer to define events that will be activated under specific conditions. Conditions are expressed in Javascript or using a custom class instance that implements ConditionI. Custom events are extremely versatile in terms of configuration, because they can be activated on a specific turn or not, and activated at the beginning or at the end of a turn. When an event is not activated on a specific turn, it can be executed several times. In order to limit the number of possible executions, a repetitions parameter can be specified, which will allow the custom event to be executed repetitions times (or no limit when repetitions = -1). The effect of an actual event can either be specified using Javascript code that will have access to the current variables value. Those values can be manipulated in the code. If Javascript is not sufficiently expressive, custom classes can be developed that will have access to every variable in the simulation. It should be noted that values should be manipulated with care because this can cause potential inconsistencies in the simulation.

Figure 29 shows a typical configuration for a custom event. In this case, an instance of the Insurgents Strategy Execution class is created and evaluated at the end of every turn. This class implements a strategy for the insurgents; it tries to evaluate the state of the situation on the next turn without the blue force intervention, and chooses the red intervention that will be the most favourable for the insurgents side. Therefore, variables values are manipulated by the custom code execution.

As for the scenario editor, a custom event is added by clicking on the Add button and removed by clicking on the Remove button. More details about every single parameter associated to custom events are provided in Table 9.



Code		☐
Code.custom	ca.lti.image_sce.scenariov2.coin2.hooks.impl.InsurgentsStrategyExecution	☑
Turn		☐
Condition	true;	☑
Condition.custom		☐
Text		☐
Before Turn	false	☑
Repetitions	-1	☑
Order		☐
VisibleTo		☐

Figure 29 - Custom events parameters

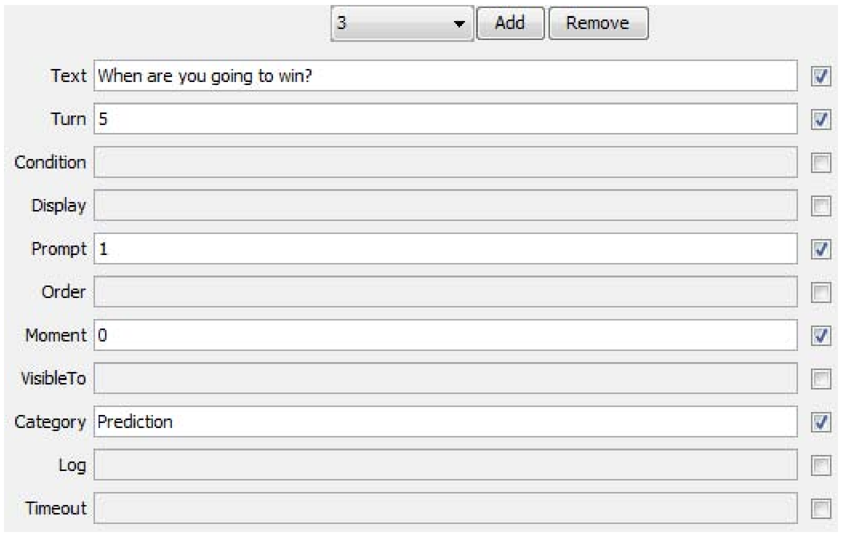
Table 9 - Details of the Custom Events parameters

Name	Type	Valid values	Description
Code	String	Any valid Javascript statement	The Javascript statement manipulates variables and has access to every instantiated variable, referenced with their ID. Sample code: variable1 = variable1 + 4
Code.custom	String	Any valid class name that implements ExecutionI	An implementation of ExecutionI allows for more sophisticated variables manipulation that cannot be expressed with simple Javascript code. Custom code has priority over normal code
Turn	Integer	-1, ...	The turn in which the custom event will be executed. -1 when turn number does not matter (default)
Condition	String	Any valid Javascript statement that returns a Boolean	The Javascript statement has access to every instantiated variable, referenced with their ID, and should return a Boolean. When True, the custom event is executed; when False, nothing happens
Condition. custom	String	Any valid class name that implements ConditionI	An implementation of ConditionI allows for more sophisticated conditions specification that cannot be expressed with simple Javascript code. Custom condition has priority over normal condition
Text	String	Any valid String	Information text that is displayed to users when the custom event is executed. By default, does not show any text. The text is shown to every user
Before Turn	Boolean	True, False	True: the custom event will be executed before the turn; False: the custom event will be executed after the turn (default)
Repetitions	Integer	-1, ...	When no turn is specified, the code can be executed several times. This parameter limits the number of repetitions. -1 for no limit
Order	Integer	Any valid Integer	When several custom events in the same turn, this parameter sets the order in which they will be evaluated/executed. Lower values are ordered first
VisibleTo	List<String>	Any valid list of Strings	List of participant Ids to which the text, if not empty, will be shown. Default: [all]

3.5 Description of Information Text

CODEM can notify users about what is happening in the situation or create prompts that request player input via Information Text. Corresponding data structures can be specified in the Info Text tab of the scenario editor. They include information messages that are shown to the user, or text prompts in which the participant has to enter a value. After being shown to the user as a popup, the text can be added (or not) to the appropriate category of a text pane which remains accessible at any time.

Information text is typically configured in the scenario editor, as shown in Figure 30. In this example, a prompt will be shown to the user with the question When are you going to win? at the beginning of turn 5 and will be added to the Prediction category in the info tab. More details on the signification of every parameter can be found in Table 10.



Text	When are you going to win?	✓
Turn	5	✓
Condition		
Display		
Prompt	1	✓
Order		
Moment	0	✓
VisibleTo		
Category	Prediction	✓
Log		
Timeout		

Figure 30 - Info text parameters

Table 10 - Details of Info Text parameters

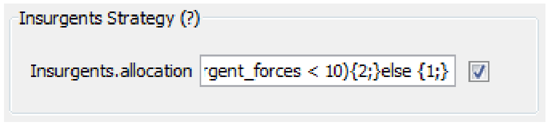
Name	Type	Valid values	Description
Text	String	Any valid String	The text that is shown to the user
Turn	Integer	-1, ...	The turn at which the info text should be displayed. -1 displays the info text each turn
Condition	String	Any valid Javascript statement	A Javascript condition that has to be true in order for the info text to be displayed. Default: no condition, activated by usual mechanisms
Display	Boolean	True, False	True: shows a popup to the user before adding the text to the info tab (default); False: hides the info text popup from the user, but shows it in the info tab
Prompt	Integer	0, 1, 2	0: the info text does not prompt the user (default); 1: prompts the user for an answer and writes the text and answer to the log and info tab; 2: prompts the user for answer but only writes it to the log file
Order	Integer	Any valid Integer	When several info texts are specified at the same moment and turn, this parameter will determine their order of appearance. Lower values are sorted first
Moment	Integer	0, 1, 2, 3	0: info text is shown before turn (default); 1: info text is shown after prediction; 2: info text is shown after decision; 3: info text is shown after turn
VisibleTo	List<String>	Any valid list of	The participants to which this info text will be

		participants	available. Participants are referred to using their ID
Category	String	Any valid String	The category of the info tab in which the current info text should be added
Log	Boolean	True, False	True: adds the info text to the log (default); False: does not add the info text to the log
Timeout	Integer	0, ...	Number of milliseconds after which, following the right moment, the info text will be shown to the user. Default: “0”

3.6 Configuration of the Scenario Editor

It is possible to configure the scenario editor via an XML configuration file that defines every single parameter in terms of its name, type and tooltip text that appears when the user moves the mouse cursor over the label. The configuration file is located in “config/EditorConfig.xml” relative to the root directory of CODEM. This can be useful to change the editor’s language if desired, or rename some concepts (change turn for year or month) or expand some tooltips. Even though every single parameter can be configured via this file, it is unadvised to do so. In fact, parameters can be added in the “Scenario” tab without any consequence on the simulation. For example, in one extension that was implemented in CODEM, there was a need to add the “Insurgents strategy” parameter, which is realized with the following XML snippet placed under the “Scenario” element; the resulting user interface elements in the scenario editor are shown in Figure 31:

```
<Group name="Insurgents Strategy" tooltip="Specific to COIN scenario">
   <Param name="Insurgents.allocation" type="string" tooltip="[Specific to COIN
scenario] Strength of insurgents' strategic strike"/>
</Group>
```


Insurgents Strategy (?)

Insurgents.allocation ·gent_forces < 10){2;}else {1;} ✓

Figure 31 - Insurgents strategy parameters

It should be noted that once a parameter is added in a scenario, some Java code must handle it. Such code can be written in extensions, as described in the following section.

3.7 Extensions

CODEM allows simulating situations composed of variables that influence one another, as well as interventions that are controlled by the user. The latter can be completely accomplished without writing a single line of code. Although a certain level of flexibility is available with the use of Javascript in conditions and code in custom events, there might be situations in which more complex processing is required. For that purpose, the experimental platform contains hook interfaces that allow for loading custom classes into the platform at runtime. Such classes should be developed with care because they can introduce modifications to the dynamic properties of agents that compose a simulation.

Conditions and executions that can accomplish more sophisticated processing for specific uses have access to the game history and the current situation. Extensions can be defined in two particular areas: winning & losing conditions and custom events (condition and execution). The two interfaces that can be implemented are:

- `ca.lti.image_sce.scenariov2.coin2.hooks.ConditionI`
    - `void init(Map<String, Object> parameters);` → Initialises the custom condition. parameters provides a set of static scenario parameters;
    - `boolean evaluate(GameHistory history, Map<String, Object> parameters, DynamicParametersContainerI dynamicParams, RandomI random);` → Evaluates the custom condition and returns true when the condition is met, otherwise false. history provides the entire game history (variables values and decisions), parameters provides a set of static scenario parameters, dynamicParams provides a set of dynamic parameters (parameters that can be modified during the simulation) and random provides a random number generator.
- `ca.lti.image_sce.scenariov2.coin2.hooks.ExecutionI`
    - `void init(Map<String, Object> parameters);` → Initialises the custom execution. parameters provides the set of static scenario parameters;
    - `Map<String, Integer> evaluate(GameHistory history, Map<String, Object> parameters, DynamicParametersContainerI dynamicParams, RandomI random);` → Evaluates the custom execution. history provides access to the game history, parameters provides a set of static scenario parameters, dynamicParams provides a set of dynamic parameters (parameters that can be modified during the simulation) and random provides a random number generator. Returns a map of relative changes that should be applied on the variables values. Simulation agents (variables and action points) can also be modified directly using static object instances; this feature should however be used with care. For example, in order to modify the number of available action points, update the unusedActionPoints dynamic property.

Another type of extension makes it possible to add participants. The implementation class should be specified in the Action Points tab of the scenario editor using the Implementation parameter. This extension should be used in cases where the participant's role is played by an external entity (e.g. learning algorithm, another implementation of the user interface). Two interfaces can be implemented in order to play the latter role:

- `ca.lti.image_sce.scenariov2.coin2.ui.dispatch.UIDispatchAdapterI`
- `ca.lti.image_sce.scenariov2.coin2.ui.dispatch.UIRemoteDispatchAdapterI`

The difference between the two interfaces is that the remote one has to implement two additional methods (“reset” and “fini”) so that several instances of the dispatcher can be started and stopped successively. Therefore, the implementer should make sure that resources are cleared when “reset” is called, and be ready for the UI dispatcher to be initialized again. When “fini” is called, no more UI will be created and the simulator will terminate. Additional details on every method that needs to be implemented are provided in Appendix C.

The next three sections describe three extensions that were implemented for demonstration purpose. The first one implements a custom execution that adds a certain level of flexibility to insurgents in a counter insurgency scenario (Section 3.7.1). The second one implements a custom winning condition that triggers when the prediction errors become stable (Section 3.7.2). The third one implements an algorithm for finding optimal solutions in a scenario (Section 3.7.3).

3.7.1 “Insurgents Strategy” Extension

This extension implements a strategy for insurgents that are able to adapt as a function of the situation. Concretely, insurgents can influence the situation by directly changing the value of one variable per turn (in addition to having other behaviours defined as mutual influences, which can change using conditional mutual influences). The extent of this influence is determined by the actual level of insurgency, and is evaluated using a Javascript statement. The context associated to the Javascript statement contains only one variable identified by the name of the insurgent forces variables; the statement should set the number of points allocated. For example, the following statement has the result of allocating 1 point if the insurgency level is smaller or equal to 6, 2 points if the insurgency level is smaller or equal to 13 and 3 points in other cases:

```
if(insurgency <= 6){1;} else if(insurgency <= 13){2;} else {3;}
```

Insurgents can directly influence the following variables: “Governance,” “Population Allegiance,” “Media,” “Infrastructures,” “Criminality,” and “Local Forces.” The direct influence is chosen by simulating what should happen in the next turn if the blue forces would not intervene in the situation, and choosing the course of action that has the most advantageous outcome for the insurgents’ side.

3.7.2 “Stable Prediction Error” Extension

This extension implements a winning condition that is triggered when a participant’s mean prediction error does not decrease anymore over a specified time window. It is hypothesized that during a mandate, the prediction error at the beginning will be high, and will decrease down to a certain level as turns pass and the participant’s understanding of the situation increases. So this is basically a stopping criterion when asymptotic anticipatory performance is achieved in the context of a function learning task.

This extension can be configured using parameters defined in the scenario editor:

- Anticipation.nbIterationsForError: this parameter specifies the time window over which the prediction error will be calculated
- Anticipation.stopCondition: this parameter specifies the stop condition expressed in Javascript that should terminate the mandate. The attributes available in the Javascript context are the following:
  - slope: the slope of prediction error over a window of Anticipation.nbIterationsForError turns
  - currentError: the mean prediction error for the current turn
  - iteration: the current iteration

Figure 32 shows how the mean prediction error should evolve as a function of turn time. The slope is defined over a time window, and should be within the level specified in the stop condition.

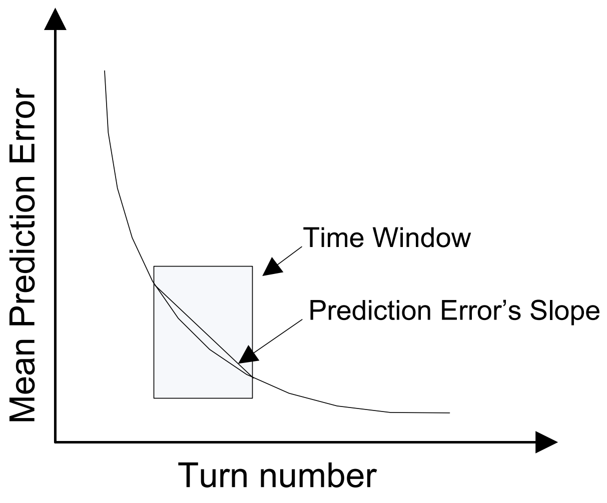


Figure 32 - Mean prediction error as a function of turn number showing the prediction error's slope in a time window

3.7.3 Genetic Algorithms for Finding Optimal Solutions

This extension implements a hook that replaces the standard user interface with an automatic decision making algorithm. The allocation of action points is realized via a genetic algorithm that optimizes a score variable. The actual genetic algorithm is implemented using the Evolver Excel add-on, which is configured to explore the decision space in an effective manner. Therefore, a software infrastructure was put in place so that CODEM can communicate with Excel in order to retrieve actions points allocation and put back the score variable. This work is still in progress, but will be further detailed once results are available. Since the hooking mechanism is very versatile, it would also allow the integration of other automatic decision algorithms such as adversaries or collaborators.

4 Conclusion

In conclusion, an experimental platform that supports the simulation of complex situations with the system dynamics formalism was developed. This platform allows experimenters to study human sensemaking and decision making in a systematic fashion. The complex situation is expressed via a scenario editor that allows modifying every single parameter straightforwardly. Those include parameters associated to variables, action points and interventions, custom events and information texts. A complex situation can be built without the need for users to write any Java code. However, the platform provides software hooks that allow for custom classes to be loaded should they be needed for more advanced use.

The experimental platform also provides a user interface that allows a user to visualize the current status of a situation, make predictions about the state of variables in the next turn, make decisions on interventions that influence the situation, take various notes, retrieve information about the past states of the situation, and observe the changes that occurred on variables during a turn. In order to facilitate the experimenter's work, log files record actions performed by the user.

5 References

Chein, M., & Mugnier, M. (2008). Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer London.

D�rner, D. (1996). The logic of failure: Recognizing and avoiding error in complex situations. Reading, MA: Addison-Wesley.

Lizotte, M., Bernier, F., Mokhtari, M., Boivin, E., DuCharme, M. and Poussart, P. (2008). Image: Simulation for Understanding Complex Situations and Increasing Future Force Agility. The 26th Army Science Conference, December 1-4, Orlando, FL.

Sterman, J.D. (2000). Business Dynamics: Systems Thinking and Modeling for a Complex World, Irwin McGraw-Hill, Boston, MA.

A. Log Files Content Description

Two types of log files are recorded during a mandate: the participant�s log and the game log. More details are available in the following sections.

A.1 Participant�s Log Description

The human log contains a trace of which actions the user performed during a turn. The file name of the human log follows this pattern: HYearMonthDay-HourMinuteSeconds-SimulationStartTime-User-ParticipantID-SessionID-GroupID

Column	Value	Description
MSG_TYPE	ACTION_POINT_AVAILABLE	The number of action points available at the beginning of a turn
SIM_TIME	Integer	The simulation time
TARGET	-	
VALUE	Integer	The number of action points
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	CONDITION_DETAILS_CLOSE	The condition details' window for the current chart has closed
SIM_TIME	Integer	The simulation time
TARGET	String	The selected condition
VALUE	String	The details about the condition
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	CONDITION_DETAILS_OPEN	The condition details' window for the current chart has opened
SIM_TIME	Integer	The simulation time
TARGET	String	The selected condition

VALUE	String	The details about the condition
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	CONFIDENCE	Confidence value about a variable has changed
SIM_TIME	Integer	The simulation time
TARGET	String	Variable name
VALUE	Integer	The confidence value
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	DECISION	An intervention changed for a variable
SIM_TIME	Integer	The simulation time
TARGET	String	Variable where more or less action points are spent
VALUE	Integer	Action points assigned to the variable
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	DECISION_DONE	The user has made his decisions for the turn
SIM_TIME	Integer	The simulation time
TARGET	-	
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	FINAL_DECISION	The final value for an intervention
SIM_TIME	Integer	The simulation time
TARGET	String	Intervention for which decision was made
VALUE	Integer	Action points assigned to the intervention
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	FINAL_PREDICTION	The final value for a prediction
SIM_TIME	Integer	The simulation time
TARGET	String	Variable for which the prediction was made
VALUE	Integer	Prediction made for variable
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	FINAL_CONFIDENCE	The final value for the confidence
SIM_TIME	Integer	The simulation time
TARGET	String	Variable for which the confidence was set
VALUE	Integer	The confidence value
LOG_TIMESTAMP	Date	Timestamp of the log message creation

MSG_TYPE	FINAL_AP_AVAILABLE	The final value of action points available
SIM_TIME	Integer	The simulation time
TARGET	Integer	Final value of action points available
VALUE	Integer	Final value of action points left
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	HIDE_VARIABLE	Variable's outgoing relations are hidden
SIM_TIME	Integer	The simulation time
TARGET	String	Variable name
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	INFO	New information added in the information tab
SIM_TIME	Integer	The simulation time
TARGET	String	If info text was a prompt, the user's answer
VALUE	String	The new information message
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	NOTE	The user added a note
SIM_TIME	Integer	The simulation time
TARGET	String	The note tab
VALUE	String	The note
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	PREDICTION	The user changed its prediction for a variable value
SIM_TIME	Integer	The simulation time
TARGET	String	Variable name
VALUE	Integer	The predicted value
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	PREDICTION_DONE	The user has made his predictions
SIM_TIME	Integer	The simulation time
TARGET	-	
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	RECEIVE_ACTION_POINT	Received a number of action points from another participant

SIM_TIME	Integer	The simulation time
TARGET	Integer	Total number of action points available
VALUE	Integer	Number of action points received
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	RELATION_DETAILS_CLOSE	Details window closed
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the relation window
VALUE	String	Details message
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	RELATION_DETAILS_OPEN	Details window opened
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the relation window
VALUE	String	Details message
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	SEND_ACTION_POINT	Send a number of action points to another participant
SIM_TIME	Integer	The simulation time
TARGET	String	The participant to which action points are sent
VALUE	Integer	Number of action points sent
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	SHOW_VARIABLE	Variable's outgoing relations are shown
SIM_TIME	Integer	The simulation time
TARGET	String	Variable name
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_ACTIVATED	A new tab is displayed
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window or frame containing the tabs
VALUE	String	Title of the tab
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_CREATED	A tab is created
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window or frame containing the

		tabs
VALUE	String	Title of the tab
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_ELAPSED_TIME	Time elapsed in a tab before going to another. It includes time spent in sub windows such as a chart
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the tab
VALUE	Integer	Elapsed time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_REMOVED	A tab is removed
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window or frame containing the tabs
VALUE	String	Title of the tab
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_TOTAL_ELAPSED_TIME	Time spent in a tab for the entire mandate
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the tab
VALUE	Integer	Elapsed time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_TOTAL_FREQUENCY	The total number of times a tab was consulted
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the tab
VALUE	Integer	Number of times the tab was consulted
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TAB_TURN_ELAPSED_TIME	Time spent in a tab for the entire turn
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the tab
VALUE	Integer	Elapsed time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TIME_DISPLAYED_CHANGE	Time displayed in the user interface
SIM_TIME	Integer	The simulation time
TARGET	String	The tab title where a displayed time change

		occurs
VALUE	Integer	The actual displayed simulation time
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TURN_DONE	The moment when next turn button is pressed
SIM_TIME	Integer	The simulation time
TARGET	-	
VALUE	Integer	Elapsed time in milliseconds since beginning of the turn
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	TURN_ELAPSED_TIME	Time elapsed for the entire turn
SIM_TIME	Integer	The simulation time
TARGET	-	
VALUE	Integer	Elapsed time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_ACTIVE_CONDITION	Change the active condition for a chart with conditions
SIM_TIME	Integer	The simulation time
TARGET	String	The condition activated
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_CLOSE	A window has closed
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_ELAPSED_TIME	Time spent in a window
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	Integer	Elapsed time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_OPEN	A window has opened
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	-	

LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_TOTAL_ELAPSED_TIME	Total time spent in a window in a mandate
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	Integer	Time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_TOTAL_FREQUENCY	Total number of times a window was open
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	Integer	Number of times
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_TOTAL_PROP_VISITED	Proportion of windows that were visited
SIM_TIME	Integer	The simulation time
TARGET	String	The type of window (or global)
VALUE	Float	Proportion of windows visited for type
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	WINDOW_TURN_ELAPSED_TIME	Total time spent in a window in a turn
SIM_TIME	Integer	The simulation time
TARGET	String	Title of the window
VALUE	Integer	Time in milliseconds
LOG_TIMESTAMP	Date	Timestamp of the log message creation

A.2 Game's Log Description

The game log only contains final values for a turn (i.e. if the user changes his predictions or decisions multiple times during a turn, only the final prediction/decision will be logged). The file name of the game log follows this pattern: GYearMonthDay-HourMinuteSeconds-SimulationStartTime

Column	Value	Description
MSG_TYPE	FILENAME	The current file name
SIM_TIME	Integer	The simulation time
VAR_NAME	-	-
VALUE	String	The current file name
USER	-	

LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	VARIABLE	Variable value
SIM_TIME	Integer	The simulation time
VAR_NAME	String	The variable name
VALUE	Integer	The variable value
USER	-	
LOG_TIMESTAMP	Date	Timestamp of the log message creation
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	PREDICTION	Variable value
SIM_TIME	Integer	The simulation time
VARIABLE	String	The variable name
VALUE	Integer	The variable predicted value
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	UNCERTAIN_VARIABLE	The variable's value has uncertainty
SIM_TIME	Integer	The simulation time
VAR_NAME	String	The variable name
VALUE	Integer	The uncertain variable's value
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	PREDICTION_ERROR	Prediction error value
SIM_TIME	Integer	The simulation time
VARIABLE	String	The variable name
VALUE	Integer	The variable prediction error value
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	CONFIDENCE	Confidence value
SIM_TIME	Integer	The simulation time
VARIABLE	String	The variable name (or global if only one)
VALUE	Integer	The prediction's confidence value
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	END_TURN	The end iteration
SIM_TIME	Integer	The simulation time

VARIABLE	-	
VALUE	-	
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation
MSG_TYPE	END_STATUS	The end of mandate status
SIM_TIME	Integer	The simulation time
VARIABLE	-	
VALUE	String	The end status (WIN, LOSE, MAX_TURN)
USER	String	The user id
LOG_TIMESTAMP	Date	Timestamp of the log message creation

A.3 Summary Logs

The summary logs contain final values for each turn in the form of a history. Six types of files are generated: variables, confidence, decisions, human, prediction error and predictions. The file name of the summary log follows this pattern: SYearMonthDay-HourMinuteSeconds-SimulationStartTime-SummaryLogType. For each summary log type, each column contains values for the corresponding turn:

- Variables: one row per variable; each column for the variable value
- Confidence: one row per variable, or only one for global confidence; each column for the confidence value
- Decisions: one row per intervention; each column for the number of action points allocated
- Human: one row for prediction, notes, relations, decision, info, situation, one row for every open window and one row for time needed to make decisions, predictions and complete a single turn; each column contains the total number of milliseconds spent at performing the corresponding operation
- Prediction error: one row per variable; each column contains the prediction error
- Prediction: one row per variable; each column contains the prediction

B. Text Labels of the User Interface

In order to create a new resource bundle, simply create a new file in the config directory with the following nomenclature: CODEMLabels_[language]_[country]_[variant]. E.g. CODEMLabels_en_US_slang

- language: a valid ISO Language Code (http://www.loc.gov/standards/iso639-2/englangn.html) (default: en)
- country: (optional) a valid ISO Country Code (http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html) or empty
- variant: (optional) anything you want, for example you could have a variant named "slang"

In order to adjust the latter properties, simply add a corresponding property to config/CODEM.properties.

Such a file should contain every single text label, as described in the following table.

Parameter Name	Description	Default Value
done	Label on the done button in decision and prediction tabs	Done
not_available	Text shown when a variable is not available in the situation tab	N/A
show	Label on the show button in the decision tab that pops up relations graph	Show
commit	Label on the commit button in the notes tab that adds new text to the current notes	Commit
default	Identifies the default relation in the relations graphs	default
info.title	Title of the info tab	Info
info.turn	Turn label in the info text prompts	Turn
info.question_abbreviation	Label that identifies a prompt question	Q
info.answer_abbreviation	Label that identifies a prompt answer	A
info.history	Title for the diagram tab in info tab	Diagram

info.default	Title for the messages tab in the info tab	Messages
info.summary	Title for the history tab in the info tab	History
info.end_of_turn	Title for the feedbacks tab in the info tab	Feedbacks
info.summary_variables	Title for the variables group in the summary tab	Variables
info.summary_decisions	Title for the decisions group in the summary tab	Decisions
note.title	Title for the notes tab	Notes
note.default_tab	Title for the default tab in the notes tab	Default
note.new_tab_name	Prompt when creating a new tab in notes tab	New tab name?
note.cannot_remove_default_tab	Notification to user that it is impossible to remove default tab	Cannot remove default tab...
note.really_remove_tab	Asks user for confirmation of removing tab	Really remove tab
note.really_commit	Asks user to really commit note when changing tab	Do you want to commit your note?
note.really_commit_title	Title for popup message to commit note	Commit note?
note.turn_number	Label for turn number mention	turn #
situation.title	Title for situation tab	Situation
situation.turn_number	Turn number shown in situation tab	Turn #
situation.next_turn	Label of the next turn button	Next turn
situation.slider_name	History time slider in the situation tab	Situation history
situation.current_value	Tooltip of the current value in the situation tab	Current value
situation.mediating_variables_legend	Legend for describing mediating variables	Light gray background = mediating variable
situation.agent_variables_legend	Legend for describing agent variables	Dark gray background = agent variable
situation.remaining_turn_time	Legend for remaining turn time	Remaining turn time
situation.remaining_decision_time	Legend for remaining decision time	Remaining decision time
relation.title	Relations tab title	Relations
relation.show_hide	Variables visibility title	Show / Hide
relation.show_all	Show all variables button label	Show all
relation.hide_all	Hide all variables button label	Hide all
relation.javascript_error	Error message when scripting engine is unavailable	Cannot init scripting engine... conditional influence will not be available
relation.line_creation_error	Error message when a relation line could not be built	Could not build line...
relation.instructions	Legend that is shown in the relations tab	Click on arrows to view relations

prediction.title	Title of the prediction tab	Prediction
prediction.legend_current	Legend shown for the current value	Current value
prediction.legend_current_abbreviation	Abbreviation of the current value text field	C
prediction.legend_prediction	Legend shown for the prediction	Prediction
prediction.legend_prediction_abbreviation	Abbreviation of the prediction value text field	P
prediction.legend_confidence	Legend shown for the confidence	Confidence
prediction.predicted_value_tooltip	Tooltip shown for the predicted value text field	Predicted value
prediction.current_value_tooltip	Tooltip shown for the current value text field	Current value
prediction.confidence_tooltip	Tooltip shown for the confidence value text field	Confidence
prediction.confidence_short	Short title for confidence radio buttons	Conf.
prediction.request_numeric	Notification when wrong value is entered in prediction text field	Please enter a numeric value...
prediction.request_between1	Notification for user entering a specific value (part1)	Please enter a value between
prediction.request_between2	Notification for user entering a specific value (part2)	and
prediction.request_between3	Notification for user entering a specific value (part3)	...
decision.title	Title for decision tab	Decision
decision.action_pts_available	Title for number of available action points	Action points available
decision.current_intervention	Title for current intervention entries	Current intervention
decision.variable_contribution_desc	Title for contribution of variables to action points	Contribution of variables to action points
decision.chart_var_on_action_pts.title1	Title for contribution chart (part1)	Contribution of
decision.chart_var_on_action_pts.title2	Title for contribution chart (part2)	on action points
decision.chart_var_on_action_pts.y_title	Y axis title for contribution chart	Contribution on action points
decision.intervention_effects	Title for effects of intervention	Effects of intervention
decision.chart_var_on_var.title1	Title for intervention effects chart (part1)	Influence of
decision.chart_var_on_var.title2	Title for intervention effects chart (part2)	on
decision.chart_var_on_var.y_title	Y axis title for intervention effects chart	Influence on
decision.slider_name	Title for decision time slider	Decision history
decision.share_AP_with	Title for action points sharing	Share AP with

decision.share_send	Label for send action points button	Send
decision.unavailable_relation	Message to the user when clicking on a relation that is not available for details	Relation details unavailable...
chart.delay	Delay label in charts	Delay
chart.condition	Condition label in charts	Condition
chart.details	Label for details button in charts	Details
chart.unavailable_condition	Label shown when a condition is unavailable	Unavailable condition
chart.active_condition	Label for indicating the active condition	ACTIVE
endmandate.variables_selection	Title in variables selection for end of mandate summary	Variables selection
endmandate.title	Title for end of mandate summary dialog	End of mandate summary
endmandate.x_title	X axis title for end of mandate dialog	Turn
endmandate.y_title	Y axis title for end of mandate dialog	Value
endmandate.all_var	Show all variables in end of mandate	All
endturn.title	Title for end of turn summary dialog	End of turn summary
endturn.changes	Title for changes end of turn summary dialog	Changes
endturn.prediction_error	Title for prediction error end of turn summary dialog	Prediction error
endturn.next	Label for next button in end of turn dialog	Next
endturn.back	Label for back button in end of turn dialog	Back
endturn.legend_new_value_abbreviation	Label for new value in prediction error	N
endturn.legend_new_value	Legend for new value in prediction error	New value
endturn.changes_instructions	Instructions for changes dialog in end of turn dialog	Move mouse over a variable to visualize details on changes
endturn.intervention_of	Title for other participantsinterventions	Interventions of
turntransition.title	Title for turn transition dialog	Turn transition
turnwait.title	Title when waiting for other participants	Waiting for other participants
main.you_win	Default message when winning	You win!!!
main.you_lose	Default message when losing	You lose!!!
main.end_mandate	Default message when end of mandate	End of mandate

C. Details of the UIDispatchAdapterI Interface

Here is the Javadoc associated to the UIDispatchAdapterI interface that one needs to implement in order to build a functional participant. Note that some are optional if you inherit from UIDispatchAdapterA. They are marked with ** sign.

```
void initLabels(java.lang.String[] actionPointLabels,
                java.lang.String[] variableIDs,
                java.lang.String[] variableLabels,
                java.lang.String[] variableDescriptions,
                java.util.Map<java.lang.String,
                   java.util.List<java.lang.String>>
                      variablesQualifiers)
```

Init the action points, variable IDs and variable labels in the main UI frame.

Parameters:

`actionPointLabels` - labels associated to every action point. The order is important

`variableIDs` - IDs associated to variables. The order is important

`variableLabels` - Labels associated to variables. The order is important

`variableDescriptions` - Descriptions associated to variables. The order is important

`variablesQualifiers` - List of variable qualifiers indexed by variable ID in a map

```
** void setMinAPValues(int[] values)
```

Sets the minimum for action points

Parameters:

`values` - the minimum action point values in the same order as action point labels

```
** void setMaxAPValues(int[] values)
```

Sets the maximum for action points

Parameters:

`values` - the maximum action point values in the same order as action point labels

```
** void initGUI(java.lang.String sideName,
             java.lang.String userID,
             java.lang.String participantID,
             java.lang.String sessionID,
             java.lang.String groupID,
             java.util.Map<java.lang.String,
                java.lang.Object> parameters,
             java.util.Collection<ca.lti.image_sce.scenariov2.
                coin2.gen.components.AP> interventions,
                 variablesInfluenceOnAP,
             java.util.List<> mutualInfluences,
             java.util.List<> conditionalMutualInfluence,
             java.util.List<java.lang.String> collaborators,
             ca.lti.image_sce.scenariov2.coin2.ui.
                callback.UIServerCallback serverCallback)
```

Initialize the main GUI.

Parameters:

`sideName` - appended to the title, which is provided by the scenario as a parameter
`userID` - the user ID
`participantID` - the participant ID, as specified at the beginning of a mandate
`sessionID` - the session ID
`groupID` - the group ID
`parameters` - the actual scenario parameters, indexed by parameter names
`interventions` - the list of interventions for the current user
`variablesInfluenceOnAP` - the provenance of action points, index by variable
`mutualInfluences` - the list of mutual influences for every variable
`conditionalMutualInfluence` - the list of conditional mutual influences for every variable.
`collaborators` - a list of collaborators with the current participant.
`serverCallback` - a reference to callback methods

`** void` **`setMaxValues`**`(int[] values)`

Set the max values for variables

Parameters:

`values` - the maximum variable values in the same order as variables IDs

---

`** void` **`setMinValues`**`(int[] values)`

Set the min values for variables

Parameters:

`values` - the minimum variable values in the same order as variables IDs

---

`** void` **`setMinGreenValues`**`(int[] values)`

Set the min values for considering a variable in the green range

Parameters:

`values` - the minimum green variable values in the same order as variables IDs

---

`** void` **`setMaxGreenValues`**`(int[] values)`

Set the max values for considering a variable in the green range

Parameters:

`values` - the maximum green variable values in the same order as variables IDs

---

`** void` **`setMinOrangeValues`**`(int[] values)`

Set the min value for considering a variable in the orange range

Parameters:

`values` - the minimum orange variable values in the same order as variables IDs

`** void `**`setMaxOrangeValues`**`(int[] values)`

Set the max value for considering a variable in the orange range

Parameters:

`values` - the maximum orange variable values in the same order as variables IDs

---

`** void `**`setUncertaintyValues`**`(int[] values)`

Set the uncertainty values for each variable

Parameters:

`values` - the uncertainty values for variables, in the same order as variables IDs

---

`** void `**`setVisible`**`(boolean visible)`

Set the main frame visible or not

Parameters:

`visible` -

`void `**`setTurn`**`(int turn)`

Set the current turn

Parameters:

`turn` - the current turn

`** void `**`setActionPoint`**`(int actionPoint)`

Set the number of available action points.

Parameters:

`actionPoint` - the number of available action points

`void `**`setVariables`**`(int[] variables)`

Set the variables' value, ordered as specified with the "initLabels" call

Parameters:

`variables` - the variables values, in the same order as variables IDs

---

`** void `**`endMandate`**`(java.util.Map<java.lang.String,java.lang.Integer> endVariablesValues, int turn, String endStatus)`

Notify the main user interface of the end of mandate, providing variables values.

Parameters:

`endVariablesValues` - variable values index by variable IDs

`turn` - the end turn

`endStatus` � the status at the end of a mandate

`** void `**`waitNextTurn`**`()`

Wait for the "next turn" button to be pressed

`int `**`getAP`**`(int index)`

Get the value of the decision made by the user for the given action point

Parameters:

`index` - the index of action point value that should be retrieved, as specified in action point labels.

Returns:

`int` **`getUnusedAP`**`()`

Returns the number of unused Action points

Returns:

`int` **`getPrediction`**`(int index)`

Get the value of the prediction made by the user for the given variable

Parameters:

`index` - the index of variable, as specified in order by variable IDs

Returns:

`int` **`getConfidence`**`(int index)`

Get the value of the confidence selected by the user for the given variable

Parameters:

`index` - - the index of variable, as specified in order by variable IDs

Returns:

```
** void          addInfoText(ca.lti.image_sce.scenariov2.coin2.gen.components.
TurnInfoText infoText, int turn)
```

Add info text with the provided parameters.

Parameters:

`infoText` -

`turn` -

```
** void initFeedbackScreen(java.lang.String[] variableIDs,
                        java.lang.String[] variableLabels,
                        java.util.List<java.lang.String>
                        disabledVariables,
                        java.util.Map<java.lang.String,
                           java.util.List<java.lang.String>>
                           variablesQualifiers,
                        java.lang.String actionPointID,
                        int endOfTurnFeedbackMode,
                        int predictionMode)
```

Initialize the end of turn feedback screen with the provided variable IDs and labels

Parameters:

`variableIDs` -

`variableLabels` -

`disabledVariables` -

`variablesQualifiers` -

`actionPointID` -

`endOfTurnFeedbackMode` -

`predictionMode` -

```
** void
waitFeedbackScreen(java.util.Map<java.lang.String,java.lang.Integer> currentV
alues,
                  java.util.Map<java.lang.String,
                  java.lang.Integer> previousValues,
                  java.util.Map<java.lang.String,
                  java.util.List<ca.lti.image_sce.scenariov2.
                  coin2.gen.components.InfluenceValue>>
                  influenceValueMap,
                  java.util.Map<java.lang.String,
                     java.lang.Integer> predictionValues,
                  java.util.Map<java.lang.String,
                     java.lang.Integer> actionPointsContributionMap)
```

Wait for the feedback screen to be closed by the user

Parameters:

`currentValues` -

`previousValues` -

`influenceValueMap` -

`predictionValues` -

`actionPointsContributionMap` -

```
** void createEndDialog(java.lang.String title,
                        java.lang.String imageFileName)
```

Create an end dialog with the given title and image file name

Parameters:

`title` -

`imageFileName` -

```
** void showEndDialog(boolean visible)
```

Show or hide the end dialog

Parameters:

`visible` -

```
** void showEndDialog(boolean visible,
                      boolean closeable)
```

Show or hide the end dialog

Parameters:

`visible` -

`closeable` - whether the end dialog is closeable

```
** void addEndSummaryVariableValues(java.lang.String label,
                                    float[] values)
```

Add variable values to the end summary dialog

Parameters:

`label` -

`values` -

```
** void showEndSummaryDialog()
```

Show the end summary dialog

```
** void createTurnTransitionDialog(java.lang.String title,
                                   java.lang.String imageFileName)
```

Create the turn transition dialog with the provided title and image file name

Parameters:

`title` -

`imageFileName` -

```
** void showTurnTransitionDialog(boolean visible,
                                  boolean fullScreen)
```
Show/hide the turn transition dialog and sets it fullscreen or not

Parameters:

`visible` -

`fullScreen` -

```
** void showTurnTransitionDialog(boolean visible)
```
Show/hide the turn transition dialog

Parameters:

`visible` -

```
** void createTurnWaitDialog(java.lang.String title,
                             java.lang.String imageFileName)
```
Create the turn wait dialog with the provided title and image file name

Parameters:

`title` -

`imageFileName` -

```
** void showTurnWaitDialog(boolean visible,
                           boolean fullScreen)
```
Show/hide the turn wait dialog and sets it fullscreen or not

Parameters:

`visible` -

`fullScreen` -

```
** void showTurnWaitDialog(boolean visible)
```
Show/hide the turn wait dialog

Parameters:

`visible` -

```
** void waitDone()
```
Wait for the user to shut down the user interface.

```
** void addActionPoints(java.lang.String fromParticipant,
                        int nbPoints)
```
Add the specified amount of action points to the current participant.

Parameters:

`fromParticipant` - the participant from which action points are sent.

`nbPoints` - the number of action points sent

D. Quick Start Reference Guide

This guide gives a quick overview on how to execute various programs that are part of the experimental platform, and describes the main operations that will lead to a functional executable simulation model.

D.1 Walkthrough

D.1.1 Single User

1) Ensure that the Java executable is in the path of your operating system (typically "bin" directory of the Java runtime environment);
2) Launch the scenario editor via "CODEM_ScenarioEditor.bat";
3) If you want to modify an existing scenario, choose "File / Open" and select the appropriate file in the file browser. Otherwise, start by adding variables and action points to your scenario. Do not forget to add instances of your variables and action points via the "Scenario" tab;
4) Save your work to an appropriate ".cdm" file;
5) Execute CODEM by double clicking on the scenario file that you just saved. If ".cdm" file extensions are not associated to CODEM, return to the scenario editor and choose "Config / Register CDM files." Double clicking on a CDM file should now work fine;
6) Once you are finished with a mandate, press "CTRL+ALT+X" in order to close the CODEM user interface.

D.1.2 Multiple Users (Cooperative or Competitive)

1) Ensure that the Java executable is in the path of your operating system (typically "bin" directory of the Java runtime environment);
2) Launch the scenario editor via "CODEM_ScenarioEditor.bat";

3) If you want to modify an existing scenario, choose �File / Open� and select the appropriate file in the file browser. Otherwise, start by adding variables and action points to your scenario. Do not forget to add instances of your variables and action points via the �Scenario� tab;
4) **For only one participant** (action point), specify �localhost� in the �Host� parameter. This participant will be referred to as �local.� **For all the others**, enter the IP address of the respective computer on which the corresponding user will be located. Those participants will be referred to as �remote�;
5) Save your work to an appropriate �.cdm� file;
6) On the computer associated to **every remote participant**, execute �CODEM_RemoteService.bat� This will enable the computer to start a user interface remotely, as requested by the local participant. There is no need to start this service on the computer associated to the local participant. Also note that once the service is launched, there is no need to start it again every time a new simulation begins;
7) On the computer associated to the local participant, double click on the �.cdm� file that contains the scenario information. If everything is configured correctly, this should launch the CODEM user interface on local computer, and one instance of the CODEM user interface on every remote computer. Otherwise, error messages will pop up indicating what went wrong;
8) Once a mandate is over, every participant has to press �CTRL+ALT+X� in order to close the CODEM user interface.

D.2 Important Files

The platform contains a hierarchy of files and directories that are of crucial importance for the platform to work correctly. Their role is the following:

- **�config� directory**: this directory contains configuration files for the scenario editor (�EditorConfig.xml�) and the variables layout (�layout.xml�). You can modify those files in order to configure the operation of the editor as well as labels in the platform;
- **�data� directory**: this directory typically contains files that configure the experimental platform (e.g. images, sound files) as well as the actual CDM simulation scenario (�GeneratedScenario.cdm�);
- **�lib� directory**: this directory contains every Jar file that supports the experimental platform. You should not directly modify files contained in this directory;
- **�logs� directory**: this directory contains log files that are associated to every single mandate;
- **�CODEM.exe�**: this file executes the experimental platform given the �GeneratedScenario.cdm� file that is currently located in the �data� directory, without displaying any output console;
- **�CODEM.bat�**: this file executes the experimental platform given the �GeneratedScenario.cdm� file that is currently located in the �data� directory, and displays an output console;
- **�CODEM_RemoteService.bat�**: this file executes the service that starts a main user interface on a remote computer. You should start it on every remote computer involved in a collaborative / competitive mandate;

- **�CODEM_ScenarioEditor.bat�**: this file executes the scenario editor. By default, the scenario editor opens an empty scenario. You can specify the file name you want to open on the command line.

D.3 Important Operations in the Experimental Platform

Once a scenario is generated by following the steps listed in the previous section, a simulation can be executed. The simulator can be started via the �CODEM.exe� file, which loads a default scenario, or by double-clicking on a CDM file. When several participants are involved in a mandate, do not forget to start �CODEM_RemoteService.bat� on every remote computer. Once the main user interface shows up, the �Situation� tab will be displayed. Depending on the game mode, other tabs will be available in the user interface. Some might be disabled, meaning that prior operations have to be completed before they can be accessed. For example, when prediction needs to be entered before making decisions, the decision tab will remain greyed out until the �Done� button has been clicked in the �Prediction� tab. In the �Relations� and �End of turn feedback� dialog, you can observe both incoming and outgoing relations. Use the right mouse button in order to switch modes.

The placement of the variables� labels can be changed through by the user. Hitting the �CTRL+E� key switches from �normal� mode to �editing� mode. In editing mode, variables� labels can be moved around the user interface by dragging them with the mouse. The main dialog can also be resized; for that purpose, simply move the mouse cursor to the edge of the dialog. A �resize� cursor will appear. Then press the left mouse button and drag the edge of the dialog to the desired size. Hitting �CTRL+E� will save the new layout and size to a file (�config/layout.xml�), which will be reloaded next time the experimental platform is executed.

In order to refrain participants from exiting the platform inadvertently, the main dialog is not closable via the well-known �X� button. Indeed, the �CTRL+ALT+X� key must be pressed in order for the platform to be closed, which will also release several resources and close open files. Log files will also be written to disk, which will be accessible in the �logs� directory. Being formatted in comma-separated values, log files can be opened in Excel or with any other software that supports this file format.

D.4 Troubleshooting of the Experimental Platform

Q1: The platform does not start when I execute the �.exe� file. What should I do?

A1: Make sure that the Java executable is in your �Path� environment variable. If it is the case but the platform still does not start correctly, run the �.bat� in the command line and look for error messages in the console.

Q2: The scenario editor does not behave as it should, what is happening?

A2: You should name the variables and action points using only letters and numbers. Please try to avoid symbols and space characters as they are not supported in XML Ids. You can keep the fancy characters to the "label" parameter.

Q3: Remote user interfaces do not show when I start the simulator. Why is that?

A3: You should first check if the Remote UI Dispatch Starter Server is running on remote computers, and if so try to restart it. You should also check whether you specified the correct IP addresses / hostnames of remote computers. If all of this is correct, try to deactivate the Windows (or Linux) firewall so that the server ports are not blocked or add an exception to your firewall configuration. Otherwise, see the console output and send the result to frioux@ltinfo.ca.

Q5: I want to build an extension, how should I proceed?

A5: It is suggested that you add your extensions in a separate Jar file. You will then not be able to use the executable file, but you can definitely use the ".bat" in order to execute the platform. Do not forget to add your .jar to the classpath. You can create your extension class by implementing appropriate interfaces and using available Java objects with the specified API, located in CODEM.jar.

E. From Conceptual Representations to System Dynamics Simulation

In order to build a generic version of the platform, tools that are part of the IMAGE project were exploited. CoGUI was used to represent a system dynamics model using agents, and the generic agent-based simulator is used as a simulation engine. CoGUI is an all-purpose knowledge representation tool. Therefore, it allows for the representation of any kind of knowledge using concepts linked by relations. However, in order for a set of graphs to be of any use, they need to be minimally structured such that a graph interpreter will be able to figure out what to generate, in terms of Java source code and XML elements, when it encounters a given concept / relation configuration. The following sections will introduce how to represent knowledge in CoGUI, and more specifically how to represent an agent-based simulation scenario. Then, the source code and XML scenario generation process will be presented for the generic agent-based simulator. The final section will introduce entities that need to be described in conceptual graphs in order to build a system dynamics simulation.

E.1 Conceptual Representation with CoGUI

Being an all-purpose knowledge representation tool, CoGUI allows for any kind of knowledge to be conveyed in a graph. The first step in building a conceptual graph with CoGUI is to define a vocabulary, or �concept type hierarchy.� The vocabulary is actually a hierarchy that defines a set of concepts that will be allowed being added to a conceptual graph. A similar hierarchy has to be defined for relations that link two or more concepts together. In Figure 33, the �relation type hierarchy� is displayed on the upper left, the �concept type hierarchy� is shown on the upper right and the instance of a conceptual graph that uses the actual relation and concept hierarchies is shown at the bottom. For more details on the notation, please refer to the CoGUI user guide.

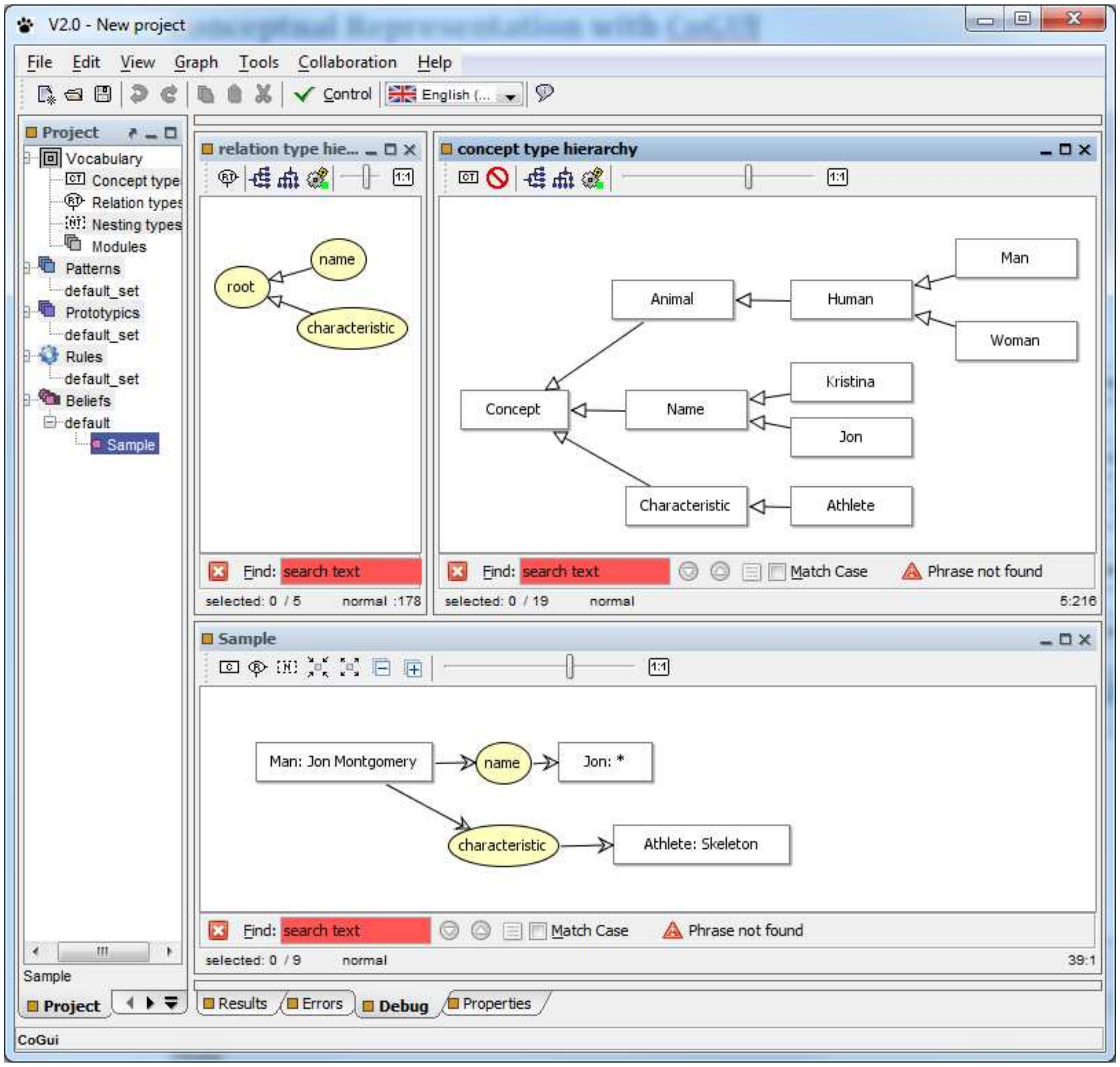


Figure 33 - Example of a conceptual graph in CoGUI

E.2 Agent-Based Simulation Using Conceptual Representations

A custom agent-based simulator was developed and integrated in IMAGE. It can simulate virtually any kind of scenario that involves autonomous entities. Depending on the role of an entity, it will be one of the following types: agent, patient or decor. Agents own static and dynamic properties and they can act on their environment through behaviours that are activated either by motivations (condition that is verified) or via reflexes (change of a dynamic property). Patients also own static and dynamic properties but they can only act on their environment through behaviours that are activated via reflexes. Decors, on the other hand, only own static properties and they cannot act on their environment nor be modified. Behaviours associated to agents and patients manipulate dynamic properties at every time step in order to update entities. This is where the actual user-written source code that defines how and when an entity will modify its environment is located.

An original contribution of IMAGE is the complete model transformation infrastructure that allows conceptual representations to be transformed to an actual executable scenario that is implemented using the custom agent-based simulator. For that purpose, an entire vocabulary and semantics was defined, resulting in conceptual graphs that can be interpreted by the scenario transformer. Figure 34 details the scenario generation process that is triggered by a user who has defined a scenario via well-constructed conceptual graphs. In order to trigger the scenario generation process, *right click* on a scenario graph located in the �Scenarios� category and choose the �*Generate scenario*� item. Note that in addition to Java skeleton classes that define entities, properties, behaviours and stimulations, an XML scenario file is built that contains values associated to properties and scenario configuration parameters. For more details on how to build a scenario using the process described in this section, please refer to the scenario generator user guide.

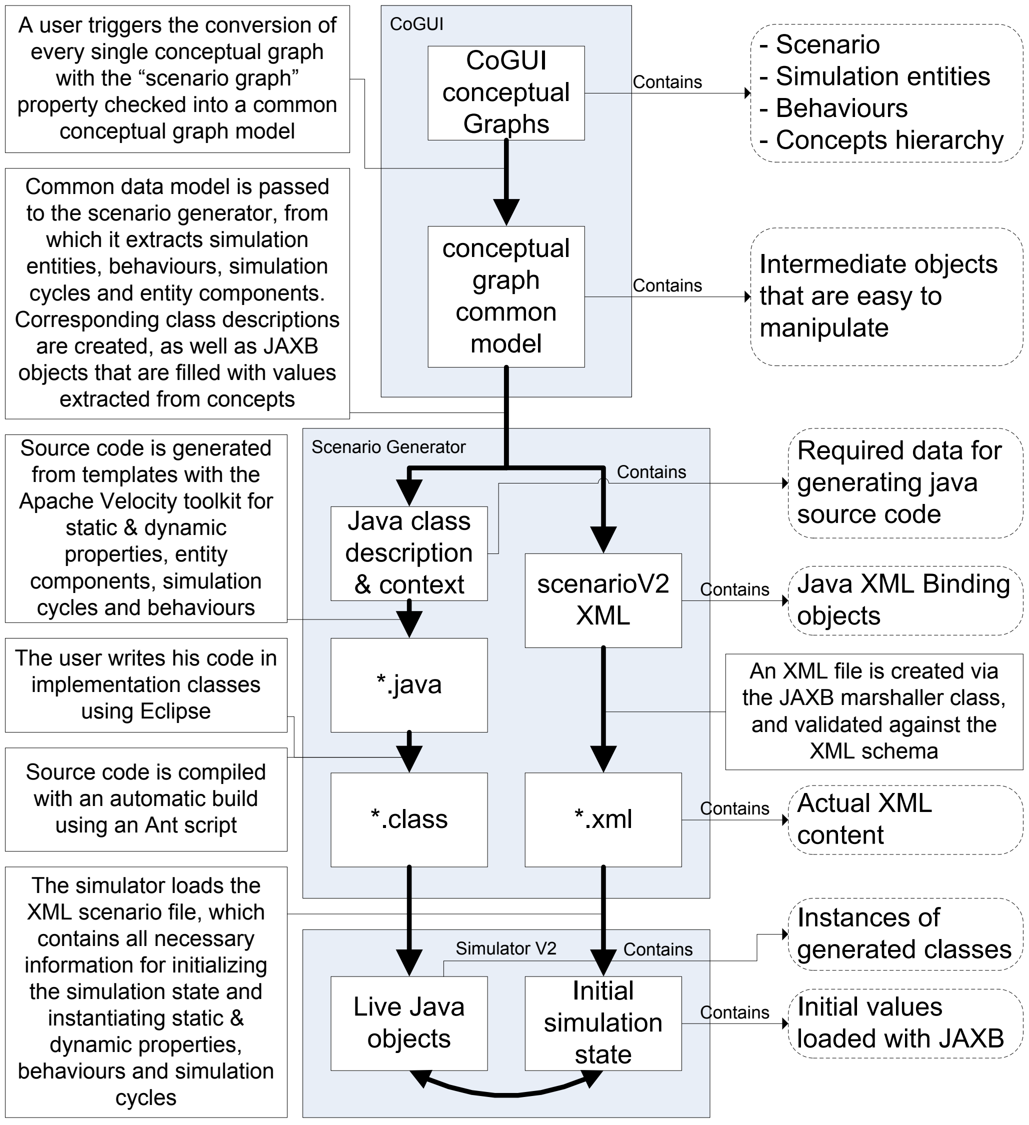

A user triggers the conversion of every single conceptual graph with the "scenario graph" property checked into a common conceptual graph model
CoGUI
CoGUI conceptual Graphs
Contains
- Scenario
- Simulation entities
- Behaviours
- Concepts hierarchy
Common data model is passed to the scenario generator, from which it extracts simulation entities, behaviours, simulation cycles and entity components. Corresponding class descriptions are created, as well as JAXB objects that are filled with values extracted from concepts
conceptual graph common model
Contains
Intermediate objects that are easy to manipulate
Source code is generated from templates with the Apache Velocity toolkit for static & dynamic properties, entity components, simulation cycles and behaviours
Scenario Generator
Contains
Required data for generating java source code
Java class description & context
scenarioV2 XML
Contains
Java XML Binding objects
The user writes his code in implementation classes using Eclipse
*.java
An XML file is created via the JAXB marshaller class, and validated against the XML schema
Source code is compiled with an automatic build using an Ant script
*.class
*.xml
Contains
Actual XML content
The simulator loads the XML scenario file, which contains all necessary information for initializing the simulation state and instantiating static & dynamic properties, behaviours and simulation cycles
Simulator V2
Contains
Instances of generated classes
Live Java objects
Initial simulation state
Contains
Initial values loaded with JAXB

Figure 34 - Scenario generation process in IMAGE

E.3 Agent-Based System Dynamics Simulation

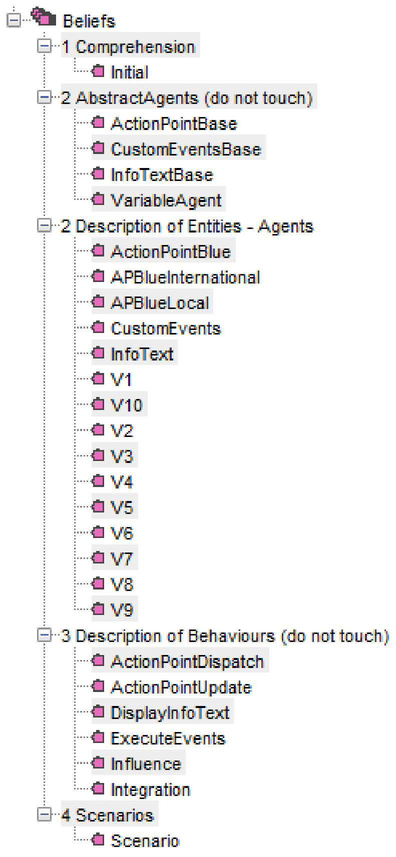
The current mandate is to develop an experimental platform that will study how humans deal with complex systems / situations. For that purpose, the generic agent-based simulator developed as part of IMAGE is used. Similar to another knowledge representation and simulation program called the Sensitivity Model (Vester, 2007), the experimental platform uses a system dynamics simulation formalism in order to perform calculations on the modeled complex system. In fact, the current platform is based on a model that contains *variables* linked by mutual influence relations, and *action points* that are allocated by a user to various interventions in order to influence the variables. One additional requirement of the experimental platform is that the user should not have to write any code in order for the resulting scenario (generated code and XML configuration file) to be functional. Fortunately, the generic agent-based simulator of IMAGE was designed such that it is possible to execute behaviours on child types, thanks to inheritance of entities types. Finally, the experimental platform must create logs of events and user behaviours in order to gain insights into the sensemaking and decision making process of participants.

At the functional level, CoGUI is started via the "CoGUI.bat" file. A set of conceptual graphs is contained in files that have the ".cogxml" extension. You can either load a complete previously built set of graphs or start of a new scenario from a template by loading the file:

"scenario/COINScenario_template.cogxml"

The latter file contains the basic conceptual structures that are needed in order to generate a fully functional complex situation scenario at the simulation level.

The set of conceptual graphs that model a complex situation is composed of several "fixed" graphs that should not be modified by a scenario developer. They are clearly identified in the graphs hierarchy that is available in CoGUI (with the "do not touch" mention). On the other hand, the graph hierarchy has to contain one "VariableAgent" graph for each defined variable, and one "ActionPoint" agent for each set of interventions that are available for a given participant. Figure 35 shows a sample hierarchy of conceptual graphs that define nine variables (V1-V9) and two potential participants (ActionPointBlue and ActionPointRed). The "AbstractAgents" category contains parent graphs that define the structure of actual agent instances, which can be *ActionPoints*, *Variables*, *CustomEvents* and *InfoText*. One graph is defined for every variable and action point, whereas all info text and custom events are added to the same graph. Moreover, instances of variables and action points are declared in the "Scenario" conceptual graph, as well as all the parameters that are used to configure the content of the actual user interface with which participants will have to interact.



```
Beliefs
    1 Comprehension
        Initial
    2 AbstractAgents (do not touch)
        ActionPointBase
        CustomEventsBase
        InfoTextBase
        VariableAgent
    2 Description of Entities - Agents
        ActionPointBlue
        APBlueInternational
        APBlueLocal
        CustomEvents
        InfoText
        V1
        V10
        V2
        V3
        V4
        V5
        V6
        V7
        V8
        V9
    3 Description of Behaviours (do not touch)
        ActionPointDispatch
        ActionPointUpdate
        DisplayInfoText
        ExecuteEvents
        Influence
        Integration
    4 Scenarios
        Scenario
```

Figure 35 - Conceptual graph hierarchy from the experimental platform

Conceptual graphs that define agentsbehaviours (category Description of Behaviours) should not be modified by the user, so is their scheduling in the scenario graph. Behaviours are configured to be executed at appropriate moments during the simulation, and they have been implemented such that a scenario designer can add as many variables and participants as he wants. The layout of the user interface changes according to the number of variables and action points, and the platform supports the distribution of participants over a local area network.

In terms of inner workings, the behaviours have the following roles:

- **ActionPointDispatch (on ActionPoint agents)**: the user interface is updated to the current variables values for a given participant and the behaviour waits until the participant clicks the next turn button;
- **ActionPointUpdate (on ActionPoint agents)**: the number of available action points is updated for a given participant;
- **DisplayInfoText (on InfoText agent)**: the info texts that need to be displayed during the current turn are retrieved and sent to the main user interface;
- **ExecuteEvents (on CustomEvent agent)**: custom events are executed if their execution condition is verified. Variables are updated accordingly;

- **Influence (on Variable agents)**: the influence of variables on other variables (delayed or not) is calculated and values are accumulated in an appropriate data structure. Variables are not modified at this time;
- **Integration (on Variable agents)**: the variable is modified according to the accumulated values calculated by the influence behaviour.

Finally, behaviours are scheduled in the following sequence:

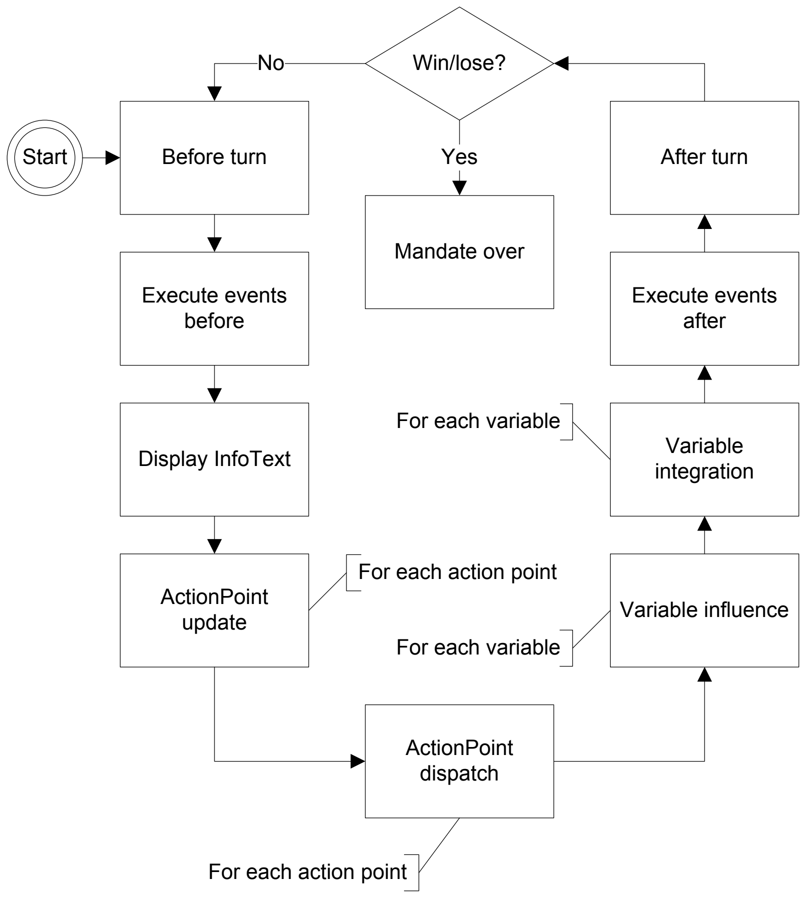


Figure 36 - Sequence of behaviours executed in the custom agent-based simulator

The sequence starts with an initial simulation state that is loaded from an XML scenario. The Before turn method is executed that is owned by the main simulation cycle. Then, behaviours that are associated to every instantiated agents are executed with the order specified in Figure 36. The After turn method, associated to the main simulation cycle, is executed after the execution of every behaviour. This includes the evaluation of winning and losing conditions. When either one of those conditions is true, the mandate is over. Otherwise, the simulation cycle continues to the next turn.

DOCUMENT CONTROL DATA
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified.)

1. ORIGINATOR (Name and address of the organization preparing the document.) LTI Informatique et génie 2700, De Carthagène Québec, Qc Canada G2B 5M4	2. SECURITY CLASSIFICATION (Overall security classification of the document, including special warning terms if applicable.) Unclassified

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

CODEM User Manual

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Roux, F., and Gagnon, J.-P.

5. DATE OF PUBLICATION (month and year of publication of document.) September 2010	6. NO. OF PAGES (Including Annexes, Appendices and DCD sheet.) 74

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract report

8a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	8b. CONTRACT NO. (If appropriate, the applicable number under which the document was written)
9a. ORIGINATOR'S DOCUMENT NUMBER (Official document number by which the document is identified by the originating activity. Number must be unique to this document.)	9b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned to this document either by the originator or the sponsor.) CR 2010-518

10. DOCUMENT AVAILABILITY (Any limitation on further distribution of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Distribution limited to defence departments
( ) Distribution limited to defence contractors
( ) Distribution limited to government
( ) Distribution limited to Defence R&D Canada
( ) Controlled by Source

11. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (10). However, where further distribution (beyond the audience specified in (10) is possible, a wider announcement audience may be selected.)

Unlimited

12. ABSTRACT (Brief and factual summary of the document. May also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). May be in English only).

13. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases characterizing a document and could be helpful in cataloguing it. Should be **Unclassified** text. If not , the classification of each term should be indicated as with the title. Equipment model designation, trade name, military project code name, and geographic location may be included. If possible, should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified.)